The Dissertation Committee for Shi Zhong
certifies that this is the approved version of the following dissertation:

Probabilistic Model-based Clustering of Complex Data

Committee:

Joydeep Ghosh, Supervisor

Alan C. Bovik

Brian L. Evans

Raymond J. Mooney

Edward J. Powers

Gerhard Werner

Probabilistic Model-based Clustering of Complex Data

by

Shi Zhong, B.S.E.E., M.S.E.E.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2003

To my mother, Wenzhen Zhou

my father, Yifu Zhong

and

my wife, Hua Hong

Acknowledgments

I thank my advisor Dr. Joydeep Ghosh for his encouragement, advice, and support throughout my Ph.D. study. With his valuable direction and help, I have been able to proceed steadily and finish my dissertation successfully. I thank the members of my dissertation committee: Dr. Alan C. Bovik, Dr. Brian L. Evans, Dr. Edward J. Powers, Dr. Raymond J. Mooney, and Dr. Gerhard Werner, for their guidance and support.

I am also grateful to a number of colleagues and friends. I thank all the current and former members of the LANS group for helpful discussions and suggestions. Especially, Dr. Kuiyu Chang, Dr. Shailesh Kumar, and Arindam Banerjee have been generous in offering discussions and asking inspiring questions.

Finally, and most important, I want to thank my parents, my parents-in-law, my wife, and my sister, for their consistent love, trust, inspiration, support, and endurance.

Shi Zhong

The University of Texas at Austin

August 2003

Probabilistic Model-based Clustering of Complex Data

Publication No. _____

Shi Zhong, Ph.D.
The University of Texas at Austin, 2003

Supervisor: Joydeep Ghosh

In many emerging data mining applications, one needs to cluster complex data such as very high-dimensional sparse text documents and continuous or discrete time sequences. Probabilistic model-based clustering techniques have shown promising results in many such applications. For real-valued low-dimensional vector data, Gaussian models have been frequently used. For very high-dimensional vector and non-vector data, model-based clustering is a natural choice when it is difficult to extract good features or identify an appropriate measure of similarity between pairs of data objects.

This dissertation presents a unified framework for model-based clustering based on a bipartite graph view of data and models. The framework includes an information-theoretic analysis of model-based partitional clustering from a deterministic annealing point of view and a view of model-based hierarchical clustering that leads to several useful extensions. The framework is used to develop two new variations of model-based clustering—a balanced model-based partitional cluster-

ing algorithm that produces clusters of comparable sizes and a hybrid model-based clustering approach that combines the advantages of partitional and hierarchical model-based algorithms.

I apply the framework and new clustering algorithms to cluster several distinct types of complex data, ranging from arbitrary-shaped 2-D synthetic data to high dimensional documents, EEG time series, and gene expression time sequences. The empirical results demonstrate the usefulness of the scalable, balanced model-based clustering algorithms, as well as the benefits of the hybrid model-based clustering approach. They also showcase the generality of the proposed clustering framework.

Contents

Chapter 1

Introduction

Clustering is a fundamental data analysis step that has been widely studied across multiple disciplines for over 40 years (Hartigan, 1975; Jain and Dubes, 1988; Jain et al., 1999; Ghosh, 2003). It has been successfully used in many exploratory pattern-analysis, grouping, decision-making, and machine-learning situations, including data mining, information retrieval, image segmentation, and pattern classification. It also has different names, such as *unsupervised classification*, *unsupervised learning*, and *segmentation*, in different communities. The general objective of a clustering problem is to discover coherent groups of similar data objects.

Though clustering problems have been well studied for decades, most traditional clustering algorithms have focused on low-dimensional vector data. More complex data such as text documents and time sequences encountered in modern data mining applications, however, pose new challenges to data clustering. Text documents are often represented as very high-dimensional sparse word vectors, whereas time sequences often contain irregular length and strong temporal dependencies and cannot be expressed efficiently as "points" in a finite dimensional vector space. Therefore, care needs to be taken when one adapts existing algorithms or designs new algorithms for emerging data mining applications.

To better understand the increasing importance of clustering techniques, let us look at several examples in modern data mining applications:

- A retailer chain company desires to segment its customers and products into meaningful groups so that effective promotional strategies can be specifically developed for each group.

- A large brokerage firm clusters stock time series into groups such that all stocks in a group have the same trends or similar underlying dynamics, in order to improve portfolio performance.

- The rapidly growing Internet easily floods Web search engines with thousands of documents in response to a query. The retrieved documents need to be automatically grouped and organized to facilitate interactive user browsing.

- The advent of DNA microarray technology makes it easy to measure the responses/expressions of thousands of genes simultaneously in an experiment. The expression data can be used to cluster genes into groups of similar functionalities, thus offering great insights into functions of unknown genes.

Clustering algorithms can be divided into *partitional* approaches and *hierarchical* approaches. A partitional method partitions data objects into a prespecified number of groups using an optimization algorithm. A hierarchical method creates a nested set of groupings and thus a structured view of data objects.

An orthogonal taxonomy of clustering algorithms favored in this dissertation is to divide existing algorithms into *pairwise* clustering and *central* clustering methods (Hofmann and Buhmann, 1997). Pairwise methods build clusters based on a data pairwise proximity measure, whereas central methods use a centroid representative for each cluster and a closeness measure between data objects and cluster centroids. These two types of methods can also be named *discriminative* approaches and *generative* approaches, or *similarity-based* approaches and *model-based* approaches, respectively. They will be compared and discussed in depth in

Chapter 2.

This dissertation deals with probabilistic model-based clustering, in which each cluster is represented by a probabilistic distribution. The basic assumption is that each data object is generated from a probabilistic model in a mixture of such models.

Gaussian model-based clustering has been researched extensively for the last two decades (Symons, 1981; McLachlan and Basford, 1988; Banfield and Raftery, 1993; Fraley, 1999). New applications and more complex data types have prompted researchers to use various other models for clustering (Smyth, 1997; Vaithyanathan and Dom, 2000; Law and Kwok, 2000; Ramoni et al., 2002; Li and Biswas, 2002). These works, however, were independently developed for different applications in which different models were used. A unified treatment is needed to better understand their relationship. One work in this direction is presented by Cadez, Gaffney, and Smyth (2000). They proposed an expectation-maximization (EM) (Dempster et al., 1977) framework for partitional clustering with a mixture of probabilistic models. Their work is basically EM clustering[1] with an emphasis on clustering irregular data (e.g., variable length sequences). It is limited in that it does not address model-based hierarchical clustering and cannot interpret well the relationship between EM clustering and other more sophisticated central clustering algorithms such as the self-organizing map (SOM) (Kohonen, 1997) and the Neural-Gas (Martinetz et al., 1993), which use a varying neighborhood function to control the assignment of data objects to different clusters. It has not been investigated before how the generalizations of SOM and Neural-Gas algorithms are related to probabilistic model-based clustering. Even the relationship between k-means (MacQueen, 1967) and EM clustering can be interpreted from different perspectives (Kearns et al., 1997; Neal and Hinton, 1998). I will further ascertain these relationships from a deterministic annealing point of view in Section 3.2.

[1]EM clustering is a specific application of the EM algorithm, for details see Section 3.2.1.

In summary, existing model-based clustering work largely focused on Gaussian models till the late nineties. Though recently many works on using different models for clustering have been successfully constructed, they were developed in isolation in different applications. No general treatment has been done so far to unify them into one framework.

The main contribution of this dissertation is a unified framework for probabilistic model-based clustering based on a bipartite graph view of data and models. The bipartite graph view provides a good understanding and visualization of existing model-based clustering algorithms, and emphasizes that the (generalized) cluster centroids are in a model space that is conceptually separate from the data space. For partitional model-based clustering, this view bears conceptual similarity to the EM algorithm. For hierarchical clustering, it points out helpful distinctions between model-based methods and similarity-based methods. The framework also includes an information-theoretic analysis of model-based partitional clustering from a deterministic annealing point of view that bridges many different algorithms such as k-means, mixture-of-models, SOM, and Neural-Gas and demonstrates the relationships among them. Extensions of model-based hierarchical clustering are presented with several newly-defined empirical Kullback-Leibler type inter-cluster distances.

The framework provides a clarifying view of existing algorithms, thus facilitates meaningful comparisons. Several generic model-based partitional and hierarchical clustering algorithms are presented using the unified framework, which encompasses a variety of existing model-based clustering algorithms as special cases. The framework is also used to develop two new variations: a balanced model-based clustering algorithm and a hybrid approach that combines the advantages of partitional and hierarchical model-based clustering algorithms. These new algorithms demonstrate the power of the unified framework: it enriches the understanding of existing algorithms in different applications and facilitates the

construction of new algorithms. The framework as well as the new algorithms are employed to cluster several distinct types of complex data, ranging from arbitrary-shaped 2-D synthetic data to high-dimensional documents, EEG time series, and gene expression time sequences. The empirical results demonstrate the usefulness of the scalable, balanced algorithms as well as the benefits of hybrid clustering approach. They also showcase the generality of the proposed model-based clustering framework.

The organization of this dissertation is as follows:

Chapter 2 provides some background on similarity-based clustering and model-based clustering, followed by a survey of existing model-based clustering methods. In addition, several commonly used probabilistic models are introduced and the deterministic annealing approach to clustering is summarized.

Chapter 3 presents the unified framework of probabilistic model-based clustering (Zhong and Ghosh, 2003c) which include a bipartite graph view of data and models, a deterministic annealing view of model-based partitional clustering, and an extension of model-based hierarchical clustering. Several generic model-based clustering algorithms, which have been instantiated with different models and used for different applications in practice, are analyzed in detail.

Chapter 4 demonstrates the application of the unified framework to document clustering (Zhong and Ghosh, 2003a). A comparative study is conducted on all the generic model-based partitional clustering algorithms discussed in Section 3.2 and instantiated with three different probabilistic models.

Chapter 5 develops a new scalable and balanced model-based clustering algorithm (Zhong and Ghosh, 2003b), based on a two-step view of model-based partitional clustering. The optimization problem for model-based k-means is decomposed into a data assignment subproblem and a model estimation subproblem. For the first subproblem, a balance-constrained version is used and solved using an efficient, approximate algorithm.

Chapter 6 shows a novel hybrid model-based clustering approach (Zhong and Ghosh, 2003c) that combines the advantages of both model-based partitional and model-based hierarchical algorithms. The usefulness of the hybrid approach is demonstrated using four case studies.

Chapter 7 summarizes this dissertation and discusses several future directions.

Chapter 2

Background and Related Work

This chapter summarizes the background needed for understanding the model-based clustering framework presented in Chapter 3. First, the motivation for model-based clustering is provided in Section 2.1 by a comparison of similarity-based clustering and model-based clustering. A summary of commonly used probabilistic models is presented in Section 2.2, followed by a discussion on existing model-based clustering algorithms in Section 2.3. Several related central clustering algorithms—deterministic annealing, self-organizing map, and neural-gas, are introduced in Section 2.4. As shown in Chapter 3, a unified treatment can be given for all these algorithms and used to provide a clarifying viewpoint. In Section 2.5, I discuss the criteria used to evaluate the quality of a clustering.

2.1 Taxonomy of Clustering Methods

Traditionally, most clustering methods are divided into *partitional* approaches and *hierarchical* approaches (Hartigan, 1975; Jain et al., 1999). A partitional method partitions data objects into K (often specified *a priori*) groups according to some optimization criterion. The widely used k-means algorithm, which will be discussed in the next section, is a classic example of a partitional method. A hierarchical method creates a set of nested clusterings, making it a good visualization

tool. Ward's algorithm, single-link, complete-link, and average-link algorithms are several widely used hierarchical clustering algorithms (Jain et al., 1999; Kamvar et al., 2002). They differ in the calculation of inter-cluster distances that are used to identify the two closest clusters to merge at each hierarchical step. For example, the single-link algorithm uses an inter-cluster distance defined as the distance between the closest pair of data objects in two clusters.

This dissertation, however, favors a taxonomy strategy recently advocated in (Banerjee and Ghosh, 2002a; Ghosh, 2003; Zhong and Ghosh, 2003b) that divides existing clustering methods into *discriminative* (or distance/similarity-based) approaches (Indyk, 1999; Scholkopf and Smola, 2001; Vapnik, 1998) and *generative* (or model-based) approaches (Blimes, 1998; Rose, 1998; Smyth, 1997). This taxonomy is orthogonal to the previous one, meaning that the clustering techniques in both similarity-based and model-based categories can be further divided into partitional and hierarchical ones.

In similarity-based approaches, such as clustering via graph partitioning (Karypis et al., 1999), one determines a distance or similarity function between pairs of data objects, and then groups similar objects together into clusters. The most commonly used distance measures are Euclidean distance and Mahalanobis distance for data that can be represented in a vector space. The instance-based learning literature (Aha et al., 1991) provides several examples of scenarios where customized distance measures perform better than such generic ones. Strehl, Ghosh, and Mooney (2000) studied the impact of different similarity measures on high-dimensional text clustering, and showed that Euclidean-type distances are not appropriate for this domain. For other complex data types (e.g., variable length sequences), defining a good similarity measure is very much data dependent and often requires expert domain knowledge. For example, a variety of distance measures have been proposed for clustering sequences (Geva and Kerem, 1998; Kalpakis et al., 2001; Eisen et al., 1998; Qian et al., 2001). Another disadvan-

tage of similarity-based approaches is that calculating the similarities between all pairs of data objects is computationally inefficient, requiring a complexity of at least $O(N^2)$, where N is the number of data objects. Despite this disadvantage, discriminative methods such as graph partitioning and spectral clustering algorithms (Karypis et al., 1999; Dhillon, 2001; Ng et al., 2002; Strehl and Ghosh, 2002) have gained recent popularity due to their ability to produce high quality, complex-shaped clusters.

Parametric, model-based approaches, on the other hand, attempt to learn generative models from the data, with each model corresponding to one particular cluster. The model type is often specified *a priori*, such as Gaussian or hidden Markov models (HMMs). The model structure (e.g., the number of hidden states in an HMM) can be determined by model selection techniques and parameters estimated using maximum likelihood algorithms, e.g., the EM algorithm (Dempster, Laird, and Rubin, 1977). Probabilistic model-based clustering techniques have shown promising results in a corpus of applications. Gaussian mixture models are the most popular models used for vector data (Symons, 1981; McLachlan and Basford, 1988; Banfield and Raftery, 1993; Fraley, 1999; Yeung et al., 2001); Multinomial models have been shown to be effective for high-dimensional text clustering (Meila and Heckerman, 2001; Vaithyanathan and Dom, 2000). For clustering more complex data such as time sequences, the dominant models are Markov Chains (MCs) (Cadez et al., 2000; Ramoni et al., 2002) and HMMs (Law and Kwok, 2000; Li and Biswas, 2000; Oates et al., 1999; Smyth, 1997). As we will see in Section 3.2.3, model-based partitional clustering algorithms have a computational complexity of $O(KNMM_1)$, where K is the number of clusters, M the number of clustering iterations, and M_1 the number of iterations for maximum likelihood model training.

In addition to the computational advantage, model-based clustering provides several other benefits. First, each cluster is described by a representative

model, which provides a probabilistic interpretation of the cluster. Second, online algorithms can be easily constructed for model-based clustering using competitive learning techniques (Banerjee and Ghosh, 2002a; Law and Kwok, 2000; Sinkkonen and Kaski, 2001). Online algorithms are useful for clustering a stream of data objects such as news feeds, as well as for incremental learning situations. Finally, probabilistic models can capture (better than pairwise similarities) the common underlying dynamics within a cluster for complex data types. For example, researchers have found that using Markov chains or hidden Markov models gives superior results for clustering time series (Cadez et al., 2000; Ramoni et al., 2002; Zhong and Ghosh, 2002b).

2.2 Commonly Used Probabilistic Models

In this section, several popular probabilistic models to be used in this dissertation are introduced. Specifically, I will introduce Gaussian models for low-dimensional vector data, multivariate Bernoulli, multinomial, and von Mises-Fisher Models for text documents, and Markov chains and hidden Markov models for time sequences.

Gaussian Models

Gaussian models are widely used for low-dimensional vector data. A Gaussian model in $\mathbb{R}^d$ is represented as $\lambda = \{\mu, \Sigma\}$,[1] where μ is the mean vector and Σ the covariance matrix. The probability density function (pdf) of a Gaussian model is

$$p(x|\lambda) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) . \tag{2.1}$$

The maximum likelihood (ML) estimation of μ and Σ is given by

$$\mu = \frac{1}{N}\sum_n x_n \ ,$$

[1]For simplicity I shall interchangeably use λ to represent a model as well as the set of parameters of that model.

and

$$\Sigma = \frac{1}{N}\sum_n (x_n - \mu)(x_n - \mu)^T \ .$$

It is easy to see that the number of parameters in the covariance matrix (Σ) grows quadratically with the dimensionality of data, which makes the model training or accurate parameter estimation for high-dimensional data a difficult task. Usually constrained Gaussian models, for which the covariance matrix is restricted to be diagonal or a constant times identity matrix, are used. The latter models are also called spherical Gaussian models. For spherical Gaussian models, we have $\Sigma = \sigma^2 I$ and thus

$$p(x|\lambda) = \frac{1}{(\sqrt{2\pi}\sigma)^d} \exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right) . \tag{2.2}$$

Unless otherwise specified, $\|\cdot\|$ represents L_2 norm.

Multivariate Bernoulli Models

Multivariate Bernoulli distributions are used to model binary vector data in $\{0,1\}^d$. Each dimension of a data vector is either 0 or 1, and is assumed to be independent of other dimensions (i.e., each dimension is an independent Bernoulli trial). The probability mass function of a multivariate Bernoulli model is

$$P(x|\lambda) = \prod_{l=1}^{d} P_l^{x^{(l)}} (1 - P_l)^{(1-x^{(l)})} \ ,$$

where $x^{(l)}$ is the l-th dimension of x, $\lambda = \{P_1, P_2, ..., P_d\}$ and P_l is the probability of $x^{(l)}$ taking value 1. The maximum-likelihood estimation of the parameters is given by

$$P_l = \frac{1}{N}\sum_n x_n^{(l)} \ .$$

This distribution has been used to model text documents and gives satisfactory results for text classification problems (McCallum and Nigam, 1998).

Multinomial Models

Multinomial probability distribution is a generalization of binomial distribution. The binomial distribution is based on Bernoulli trials in which only two outcomes are possible. The multinomial distribution is based on a generalized Bernoulli trial in which d outcomes are possible. The parameters for a multinomial distribution are $\lambda = \{P_1, P_2, ..., P_d\}$ and subject to the constraints $P_l \geq 0, \forall l$ and $\sum_l P_l = 1$. The l-th dimension of a data vector is the number of times the l-th outcome being observed. If the event sequence of a data vector x is viewed as an ordered string of outcomes, then the probability of observing x is

$$P(x|\lambda) = \prod_{l=1}^{d} P_l^{x^{(l)}} \ .$$

A different (greater) probability is obtained when an order of outcomes is not specified (Stark and Woods, 1994; McCallum and Nigam, 1998; Nigam, 2001):

$$P(x|\lambda) = \frac{M!}{x^{(1)}!x^{(2)}! \cdots x^{(d)}!} \prod_{l=1}^{d} P_l^{x^{(l)}} \ ,$$

where $M = \sum_l x^{(l)}$ is the total number of Bernoulli trials. The maximum-likelihood estimation of the parameters is given by

$$P_l = \frac{\sum_n x_n^{(l)}}{\sum_l \sum_n x_n^{(l)}} \ .$$

von Mises-Fisher Models

The von Mises-Fisher (vMF) distribution is the analogue of the Gaussian distribution for directional data in the sense that it is the unique distribution of L_2-normalized data that maximizes the entropy given the first and second moments of the distribution (Mardia, 1975). The pdf of a vMF distribution is

$$p(x|\lambda) = \frac{1}{Z(\kappa)} \exp\left(\kappa \cdot x^T \mu\right) \ ,$$

where x is a L_2-normalized data vector, μ the L_2-normalized mean vector, and the Bessel function $Z(\kappa)$ a normalization term. The κ measures the directional variance (or dispersion) and the higher it is, the more peaked the distribution is. The maximum likelihood estimation of μ is simple and given by $\mu = \frac{\sum x}{\|\sum x\|}$. The estimation of κ, however, is rather difficult due to the Bessel function involved (Banerjee and Ghosh, 2002a; Banerjee et al., 2003). In a k-means clustering setting, if κ is assumed to be the same for all clusters, then the clustering results do not depend on κ, which can be ignored. For EM clustering, Banerjee et al. (2003) worked out an approximate solution for the vMF models.

Markov Chain Models

Markov chains are widely used for modeling discrete symbolic sequences and capturing stationary temporal dependencies. Here I only discuss the first-order Markov chains, for which a first-order Markov assumption is used. That is, each observation depends only on the observation at previous time step.

A Markov chain can be characterized by $\lambda = \{\pi, A\}$, where $\pi = \{\pi_i\}_{i=1,\ldots,M}$ is a set of priors for the M observation symbols in an observation alphabet. The transition matrix $A = \{a_{ij}\}_{i,j\in\{1,\ldots,M\}}$ specifies the transition probabilities between different observation symbols from one time step to the next. Let x be a sequence of symbols and $x(t)$ the symbol at time t. The transition probability a_{ij} is defined as $a_{ij} = P(x(t+1) = j | x(t) = i), \forall t$. The probability of a sequence x given λ is

$$P(x|\lambda) = \pi_{x(1)} \prod_{t=1}^{T_x - 1} a_{x(t)x(t+1)} \ ,$$

where T_x is the length of sequence x. Given a set of sequences X, the maximum-likelihood estimation of a_{ij} is given by

$$a_{ij} = \frac{\sum_{x\in X} \sum_{t=1}^{T_x - 1} I(x(t) = i, x(t+1) = j)}{\sum_{x\in X} \sum_{t=1}^{T_x - 1} I(x(t) = i)} \ ,$$

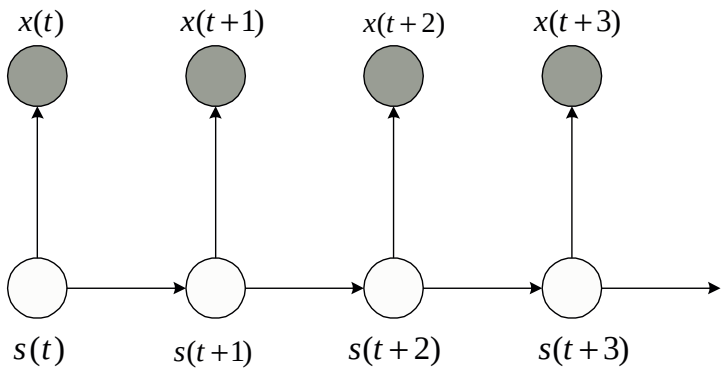


Figure 2.1: A first order HMM model. The empty circles are the hidden states and the shaded ones the observations.

where $I(.)$ is an indicator function that takes a value of 1 if the predicate argument is true and 0 otherwise.

Hidden Markov Models

HMMs have been heavily researched and used in numerous applications for the past several decades, especially in the speech recognition area (Bengio, 1999; Rabiner, 1989). A standard first-order HMM model uses a discrete hidden state at time t to summarize all the information before t and thus the observation at any time only depends on the current hidden state. The hidden state sequence is a first-order Markov chain. Such an HMM unrolled over several time slices is shown in Fig. 2.1.

A standard first-order HMM is usually denoted as a triplet $\lambda = (\pi, A, B)$. $\pi = \{\pi_i\}$ (where $\sum_i \pi_i = 1$) is the prior probability distribution of hidden states. $A = \{a_{ij}\}$ (where $\sum_j a_{ij} = 1$) is the transition probability distribution between hidden states. For the discrete observation case, the observation distribution is $B = \{b_j(k)\}$ (where $\sum_k b_j(k) = 1$). For the continuous observation case, the

observation distribution is usually modeled by a mixture-of-Gaussians

$$b_j(x) = \sum_l c_{jl} p(x|\mu_{jl}, \Sigma_{jl}), \quad \sum_l c_{jl} = 1$$

where x is the observation vector being modeled, c_{jl} the mixture weight, μ_{jl} the mean vector of the m-th mixture, Σ_{jl} the covariance matrix of the l-th mixture for state j. The likelihood of observing a sequence x is

$$P(x|\lambda) = \sum_s \pi_{s(1)} \prod_{t=1}^{T-1} a_{s(t)s(t+1)} \prod_{t=1}^{T} b_{s(t)}(x(t)) \ , \tag{2.3}$$

where s is a hidden state sequence and T the sequence length. Let M be the number of hidden states. There are a total of M^T different hidden state sequences, making the calculation of (2.3) a nontrivial task. Usually a dynamic programming method called forward-backward procedure (Rabiner, 1989) is employed to solve the problem efficiently, in time $O(TM^2)$.

The most well-known training algorithm for HMMs is the Baum-Welch algorithm (Baum, 1969; Rabiner, 1989), which is a maximum likelihood approach. Juang et al. (1986) pointed out that using mixture-of-Gaussians as the observation model of HMM sometimes results in singularity problems during the ML training, i.e., the likelihood $P(x|\lambda)$ goes to infinity. They suggested solving the problem by restarting from a different (random) initialization. Another popular way of dealing with the singularity problem is to use maximum *a priori* (MAP) learning instead of ML learning. An MAP learning algorithm maximizes $P(\Lambda|X) \propto P(X|\Lambda)P(\Lambda)$, where $P(\Lambda)$ is the model prior and needs to be specified properly. It has been widely recognized that the use of proper parameter priors smoothes the search space of an optimization problem and makes the learning algorithm more stable. Gauvain and Lee (1994) provided a fairly complete description of the MAP learning algorithms for continuous HMMs.

Recently, HMMs have been extended to solve various time-series and sequence analysis problems, such as complex human action recognition (Brand et al.,

1997), protein sequence modeling (Eddy, 1998), traffic modeling (Kwon and Murphy, 2000) and biosignal analysis (Rezek and Roberts, 2000). It has also been shown that (improved) HMM models perform very well for EEG time series classification (Zhong and Ghosh, 2002a).

2.3 Model-based Clustering Algorithms

Model-based Partitional Clustering

Let us start the discussion with the widely used standard k-means algorithm (MacQueen, 1967), shown in Fig. 2.2. Obviously, this algorithm belongs to the central clustering category. The objective function it minimizes is

$$E_{\mathrm{k-means}} = \sum_n \|x_n - \mu_{y_n}\|^2 \ , \tag{2.4}$$

where $y_n = \arg\min_k \|x_n - \mu_k\|^2$ is the cluster identity of data vector x_n and μ_{y_n} is the centroid of cluster y_n.

A question that might immediately come to the reader's mind is, "why is k-means a model-based clustering algorithm?" This is because the k-means algorithm is often seen as a special, limiting case of the mixture-of-Gaussians clustering, with each Gaussian model being spherical (i.e., the covariance matrix is a constant times identity matrix) and having identical covariance matrix. This also brings up another widely used algorithm—the EM algorithm (Dempster et al., 1977), which can be used to solve any maximum likelihood optimization problems with hidden variables or missing data. The EM algorithm iterates between an E-step and an M-step until convergence. In the E-step, a posterior probability distribution on the hidden variable(s) is estimated. In the M-step, the estimated posterior probabilities are used to estimate all other parameters. The EM algorithm itself is not tied up with clustering but it is commonly used for the mixture

Algorithm: k-means

Input: A set of N data vectors $X = \{x_1, ..., x_N\}$ in $\mathbb{R}^d$ and the number of clusters K.

Output: A partition of the data vectors given by the cluster identity vector $Y = \{y_1, ...y_N\},\ y_n \in \{1, ..., K\}$.

Steps:

1. Initialization: initialize the cluster centroid vectors $\{\mu_1, ..., \mu_K\}$;

2a. Data assignment: for data vector x_n, set $y_n = \arg\min_k \|x_n - \mu_k\|^2$;

2b. Centroid calculation: for each cluster k, let $X_k = \{x_n | y_n = k\}$, the centroid is estimated as $\mu_k = \frac{1}{|X_k|} \sum_{x \in X_k} x$;

3. Stop if Y does not change, otherwise go back to Step 2a.

Figure 2.2: Standard k-means algorithm.

of probabilistic models clustering (Cadez et al., 2000), which is often referred to as *EM clustering*. I will use this terminology throughout this dissertation. The mixture-of-Gaussians clustering using EM algorithm is shown in Fig. 2.3. A detailed derivation of the parameter estimation for the mixture-of-Gaussians as well as an excellent tutorial on the EM algorithm can be found in (Blimes, 1998).

The mixture-of-Gaussians clustering is also called "soft" clustering because the E-step is equivalent to using "soft" assignment of data vectors to clusters (i.e., each data vector is fractionally assigned to multiple clusters). This is in contrast to the k-means algorithm, in which the data assignment is considered "hard". Though it is widely accepted that the k-means algorithm is a special case of the mixture-of-Gaussians clustering algorithm, it is not straightforward to derive the k-means algorithm from the mixture-of-Gaussians clustering. A common derivation is to first set the covariance matrix to be σI, i.e., using spherical Gaussian

Algorithm: mixture-of-Gaussians clustering

Input: A set of N data vectors $X = \{x_1, ..., x_N\}$, model structure $\Lambda = \{\mu_k, \Sigma_k, \alpha_k\}_{k=1,...,K}$, where μ's and Σ's are the parameters for Gaussian models and α's are prior parameters that are subject to $\alpha_k \geq 0, \forall k$ and $\sum_k \alpha_k = 1$.

Output: Trained model parameters Λ that maximizes the data likelihood

$$P(X|\Lambda) = \prod_n \sum_k \alpha_k p(x_n|\lambda_k) \ ,$$

and a partition of the data vectors given by the cluster identity vector $Y = \{y_1, ..., y_N\}, \ y_n \in \{1, ..., K\}$.

Steps:

1. Initialization: initialize the model parameters Λ;

2a. E-step: the posterior probability of model k, given a data vector x_n and current model parameters Λ, is estimated as

$$P(k|x_n, \Lambda) = \frac{\alpha_k p(x_n|\lambda_k)}{\sum_j \alpha_j p(x_n|\lambda_j)},$$

where the pdf $p(x|\lambda)$ is given in (2.1);

2b. M-step: the maximum likelihood re-estimation of model parameters Λ is given by

$$\mu_k^{(new)} = \frac{\sum_n P(k|x_n, \Lambda) x_n}{\sum_n P(k|x_n, \Lambda)},$$

$$\Sigma_k^{(new)} = \frac{\sum_n P(k|x_n, \Lambda)(x_n - \mu_k)(x_n - \mu_k)^T}{\sum_n P(k|x_n, \Lambda)},$$

and

$$\alpha_k^{(new)} = \frac{1}{N} \sum_n P(k|x_n, \Lambda) \ ;$$

3. Stop if $P(X|\Lambda)$ converges, otherwise go back to Step 2a;

4. For each data vector x_n, set $y_n = \arg\max_k (\alpha_k p(x_n|\lambda_k))$.

Figure 2.3: Mixture-of-Gaussians clustering algorithm.

models (2.2), and set α's to be the same[2] $(1/K)$, and then let the variance σ go to 0^+. In the limit, the posterior probability $P(k|x_n, \Lambda)$ becomes either 1 or 0. That is, the posterior data assignments become hard and the mixture-of-Gaussians clustering becomes the k-means algorithm (Mitchell, 1997). This interpretation, however, does not work for the generalized k-means and EM clustering algorithms that will be discussed in Chapter 3. For example, when a discrete probabilistic model is used, there is no parameter (such as the σ in a Gaussian) associated with the model that can be tuned to let $P(k|x_n, \Lambda)$ go to 0 or 1. In the next chapter, I will discuss a better interpretation from a deterministic annealing point of view. The deterministic annealing algorithm is introduced in Section 2.4.

The Gaussian model-based clustering algorithms have been analyzed in depth and successfully applied in many applications (Banfield and Raftery, 1993; Fraley and Raftery, 1998; Fraley, 1999; Yeung et al., 2001). For example, Yeung et al. (2001) recently applied the Gaussian-based clustering to gene expression data and claimed that it produced satisfactory results.

Extended k-means and EM clustering algorithms have been applied to non-Gaussian models for clustering more complex data. For very high-dimensional text clustering, multinomial model-based clustering methods have been proposed. For instance, Meila and Heckerman (2001) compared several multinomial model-based clustering methods on text data. They also worked out an efficient hierarchical multinomial model-based clustering algorithm that has a complexity of between $O(N^2)$ and $O(N^3)$. The spherical k-means algorithm (Dhillon and Modha, 2001; Dhillon et al., 2001), for clustering large document collections, can be derived from a generative model based on the vMF distribution under certain restrictive conditions (Banerjee and Ghosh, 2002a; Banerjee et al., 2003).

For time sequence data, generative model-based clustering approaches are

[2]Actually it can be easily shown that this is not necessary for the reduction from mixture-of-Gaussians to k-means.

a natural fit, given the difficulty of constructing feature vectors or finding good similarity measures between sequences. The most popular models used for time series are Markov Chains (Cadez et al., 2000; Ramoni et al., 2002) and HMMs (Smyth, 1997; Oates et al., 1999; Law and Kwok, 2000; Li and Biswas, 2000). The use of HMMs for clustering sequences appears to have been mentioned first in (Juang and Rabiner, 1985) and subsequently used in the context of discovering sub-families of protein sequences in (Krogh, 1994). Smyth (1997) introduced a sequence clustering algorithm with HMMs and a Bayesian model selection method for determining the number of clusters. Oates et al. (1999) combined dynamic time warping and HMMs for clustering time series data. Law and Kwok (2000) proposed Rival Penalized Competitive Learning for HMM-based sequence clustering. Their method is really an online version of the HMM-based k-means algorithm. Li and Biswas (2000) extended the Bayesian model selection method to HMM-based sequence clustering and showed that the Bayesian Information Criterion can help find the number of clusters and appropriate HMM structures.

Model-based Hierarchical Clustering

For partitional clustering methods, the number of clusters needs to be specified *a priori*. This number, however, is often unknown in many clustering problems. Moreover, sometimes one prefers the clustering algorithm to return a series of nested clusterings for interactive analysis (Seo and Shneiderman, 2002). Hierarchical clustering techniques provide such an advantage. Although one can run k-means or mixture-of-models clustering multiple times with different numbers of clusters, the returned clusterings are not guaranteed to be structurally related. Bottom-up hierarchical agglomerative clustering (HAC) has been the most popular hierarchical method, although top-down methods have also been used, e.g., Steinbach et al. (2000).

Although in special cases the complexity of HAC algorithms can be reduced to $O(N^2)$ for Gaussian-based hierarchical clustering as shown in (Banfield and Raftery, 1993) and (Fraley, 1999), in general the complexity is at least $O(N^2 \log N)$ and higher than that of partitional methods (Jain et al., 1999). For example, Ramoni et al. (2002) applied the HAC approach with MC models to cluster robot sensor data and the computational complexity of their algorithm is $O(N^4 N_s^2)$, where N_s is the number of states in the MC models.

The majority of model-based clustering methods are based on the maximum likelihood formulation (Symons, 1981; McLachlan and Basford, 1988; Banfield and Raftery, 1993), even for hierarchical ones (Fraley, 1999; Vaithyanathan and Dom, 2000; Kamvar et al., 2002). Due to high computational complexity, hierarchical model-based clustering is not popular and early works focused on deriving efficient hierarchical algorithms for special cases of Gaussian models (Fraley, 1999). Meila and Heckerman (2001) derived efficient multinomial model-based hierarchical clustering algorithms, but only ran it on a sampled dataset to provide an initialization to a subsequent multinomial model-based EM clustering step. Vaithyanathan and Dom (2000) built multinomial model-based hierarchical clustering on top of the clustering results obtained from a partitional clustering, in order to save computational efforts. Their work motivated the hybrid model-based clustering work in Chapter 6.

2.4 Deterministic Annealing for Clustering

Simulated annealing (Kirkpatrick et al., 1983) is a stochastic optimization technique motivated by the annealing processes in physical chemistry. Certain chemical systems can be driven to their low-energy states by annealing, which is a gradual reduction of temperature. The annealing schedule, i.e., the rate at which the temperature is lowered, is critical to reaching the global optimum. Geman and

Geman (1984) have theoretically shown that the global optimum can be achieved by a schedule following $T \propto \frac{1}{\log m}$, where m is the current iteration number. Such schedules are very slow and unrealistic in many applications.

Deterministic annealing (DA), as the name suggests, is a deterministic version of simulated annealing. It is derived from an information theoretic view of optimization problems. It is not guaranteed to reach global optimum; rather it is a heuristic strategy that avoids many local solutions and enjoys a faster temperature schedule. It has been applied to clustering, regression, classification, and many other applications (Rose, 1998). In this section, I simply summarize the derivation of clustering via deterministic annealing.

Consider a central clustering problem, in which X is a set of source data vectors, Y is a set of codebook (cluster centroid) vectors. The objective is to minimize the average distortion

$$E = \sum_x P(x) d(x, y(x)) \tag{2.5}$$

where $y(x) = \arg\min_y d(x, y)$ is the codebook vector for x and $d(x, y)$ the distortion resulting from representing x by y. The source data priors $P(x)$'s are usually approximated by $1/N$ in practice. From a vector quantization point of view, x's are quantized into y's in such a way that the average distortion is minimized. Minimizing (2.5) leads to the well known Lloyd algorithm (Lloyd, 1982). Similar algorithms proposed in pattern recognition literature are the ISODATA algorithm (Hall and Ball, 1967) and the k-means algorithm (MacQueen, 1967). These algorithms perform "hard" clustering in the sense that each data vector is associated with only one cluster (the cluster with the smallest distortion) at each iteration of the clustering process. To derive deterministic annealing, let us introduce "soft" assignment, in which a data vector can be probabilistically associated with multiple cluster centroids. The soft association can be governed by the joint probability distribution $P(x, y)$.

With soft assignment, the cost function for clustering is the expected distortion

$$E_1 = \sum_{x,y} P(x,y)d(x,y) = \sum_x P(x) \sum_y P(y|x)d(x,y) \tag{2.6}$$

Directly minimizing E_1 over $P(y|x)$ leads to hard assignment, i.e., $P(y|x)$ is 1 for $y = \arg\max_{y'} d(x,y')$ and 0 otherwise. To enforce a certain randomness of the assignment during the optimization process, an entropy-constrained version of the cost function is

$$F = E_1 - T \cdot H(X,Y) = E_1 - T \cdot H(Y|X) - T \cdot H(X) \ , \tag{2.7}$$

where

$$H(Y|X) = -\sum_x P(x) \sum_y P(y|x) \log P(y|x)$$

is the conditional entropy for Y and T is a Lagrange multiplier. Note that the data prior entropy $H(X)$ is usually treated as a constant. Minimizing F can also be interpreted as maximum entropy clustering: F has a free energy interpretation in statistical physics and T can be interpreted as a temperature. Suppose the expected distortion E_1 is fixed, minimizing F is equivalent to maximizing the entropy $H(Y|X)$. For this reason, the clustering algorithm minimizing F is also called maximum entropy clustering. It is well known that the maximum entropy solution for $P(y|x)$ is the Gibbs distribution

$$P(y|x) = \frac{\exp\left(-\frac{d(x,y)}{T}\right)}{\sum_{y'} \exp\left(-\frac{d(x,y')}{T}\right)} \ ,$$

which is parameterized by the temperature parameter T. One can add one more entropy constraint to F

$$F_1 = F + \gamma H(Y) \ , \tag{2.8}$$

where $H(Y) = -\sum_y P(y) \log P(y)$ is the prior entropy for Y, and γ is a Lagrange multiplier. One intuition behind keeping $H(Y)$ low is to use a smaller number of

codebook vectors and thus to lower the prior uncertainty. This formulation has an entropy-constrained vector quantization interpretation (Rose, 1998). One interesting case is when $\gamma = T$, minimizing F_1 leads to the so-called mass-constrained clustering (Rose et al., 1993; Rose, 1998) and F_1 can be rewritten as

$$F_1 = E_1 + T \cdot I(X;Y) \ . \tag{2.9}$$

Now it is clear that another possible interpretation on minimizing F_1 is to minimize the mutual information between X and Y, i.e., to compress X into Y as much as possible, and meanwhile to keep the distortion low. The temperature parameter T is used to adjust such a tradeoff. This special case also has a good interpretation in the model-based clustering context, as shown in the next chapter. From this point forward, I only consider the $\gamma = T$ case.

Minimizing F_1 in (2.9) leads to the following optimality conditions:

$$P(y|x) = \frac{P(y) \exp\left(-\frac{d(x,y)}{T}\right)}{\sum_{y'} P(y') \exp\left(-\frac{d(x,y')}{T}\right)} \ , \tag{2.10}$$

and

$$\sum_x P(y|x) \frac{\partial}{\partial y} d(x,y) = 0 \ . \tag{2.11}$$

They can be solved by the EM algorithm in an iterative fashion. The computational annealing process starts with a high temperature and slowly decreases to zero. A detailed clustering procedure via deterministic annealing was given by Rose (1998), who also demonstrated the superiority of deterministic annealing over a wide range of applications. For simplicity, later I use *DA clustering* to refer to the clustering algorithm via deterministic annealing.

Though derived for central clustering, deterministic annealing has been fruitfully applied to pairwise clustering (Hofmann and Buhmann, 1997). In this dissertation, I will focus on using deterministic annealing to interpret probabilistic model-based partitional clustering.

Before leaving this section, I briefly introduce two other related clustering algorithms that have an annealing flavor—the self-organizing map (Kohonen, 1997) and the Neural-Gas clustering (Martinetz et al., 1993), both from the competitive learning literature. For a better comparison, I only describe the batch versions of the two algorithms here.

A distinct feature of SOM is the use of a topological map, in which each cluster has a fixed coordinate. Let the map location of cluster k be ϕ_k and $K_\alpha(\phi_1, \phi_2) = \exp\left(-\frac{\|\phi_1-\phi_2\|^2}{2\alpha^2}\right)$ a neighborhood function. Recall that $y(x) = \arg\min_y d(x, y)$. The batch SOM algorithm amounts to iterating between the following two steps:

$$P(y|x) = \frac{K_\alpha(\phi_y, \phi_{y(x)})}{\sum_{y'} K_\alpha(\phi_{y'}, \phi_{y(x)})} , \tag{2.12}$$

and

$$\mu_y = \frac{1}{N} \sum_x P(y|x)x , \tag{2.13}$$

where α is a parameter controlling the width of the neighborhood function and decreases gradually during the clustering process. It is immediately noted that the α has the same functionality of a temperature parameter in deterministic annealing. The difference is that here the calculation of $P(y|x)$ is constrained by a topological map structure, which gives SOM the advantage that all resulting clusters are structurally related according to the pre-specified topological map.

The Neural-Gas algorithm differs from the SOM and DA clustering, only in how $P(y|x)$ is calculated

$$P(y|x) = \frac{e^{-r(x,y)/\beta}}{\sum_{y'} e^{-r(x,y')/\beta}} , \tag{2.14}$$

where β is an equivalent temperature parameter and $r(x, y)$ a rank function that takes value $k - 1$ if y is the k-th closest cluster centroid to data vector x. It has been shown that an online version of this algorithm can converge faster and find better local solutions than the SOM and DA clustering for certain problems (Martinetz et al., 1993).

Deterministic annealing has been successfully used in a wide range of applications Rose (1998). But its applications to clustering have been largely restricted to vector data (Rose et al., 1993; Hofmann and Buhmann, 1997). Hofmann and Buhmann (1998) provided a unified treatment of SOM, Neural-Gas, and deterministic annealing algorithms for vector quantization applications (Gersho and Gray, 1992). They showed that the three types of algorithms are three different implementations of a continuation method (Allgower and Georg, 1990) for vector quantization, with different competitive learning rules. None of these work, however, have analyzed probabilistic model-based clustering or demonstrated the relationship between k-means and EM clustering from an annealing perspective.

2.5 Clustering Evaluation

In the section, I will discuss how the quality of a clustering can be evaluated and the criteria used to compare different clustering algorithms.

Subjective (human) evaluation is often difficult and expensive, yet is still indispensable in many real applications. Objective evaluation criteria include intrinsic measures and extrinsic measures (Jain et al., 1999). Intrinsic measures formulate quality as a function of the given data and similarities/models and are often the same as the objective function that a clustering algorithm explicitly optimizes. For example, the data likelihood objective was used by Meila and Heckerman (2001) to cluster text data using multinomial models. For low-dimensional vector data, the average (or summed) distance from cluster centers, e.g., the sum-squared error criteria used for the standard k-means algorithm, is a common criterion.

Extrinsic measures are commonly used when the category labels of data are known (but of course not used in the clustering process). Examples of external measures include the confusion matrix, classification accuracy, F1 measure,

average purity, average entropy, and mutual information (Ghosh, 2003). Other statistical measures are also available, such as the Rand index (Rand, 1971) and Fowlkes-Mallows measure (Fowlkes and Mallows, 1983).

F1 measure is often used in the information retrieval communities, where clustering serves as a way of improving the quality and accelerating the speed of search. The purity of a cluster is defined as the percentage of the majority category in the cluster. Entropy $H(\cdot)$ measures the category spread or uncertainty of a cluster and can be normalized to the range $[0, 1]$ by dividing $\log K$, where K is the number of classes. If all objects in a cluster come from one category, the purity is 1 and the normalized entropy is 0. If a cluster contains an equal number of objects from each category, the purity is $1/K$ and the normalized entropy is 1.

It has been argued that the mutual information $I(Y;\hat{Y})$ between a *r.v.* Y, governing the cluster labels, and a *r.v.* $\hat{Y}$, governing the class labels, is a superior measure to purity or entropy (Strehl and Ghosh, 2002). Moreover, by normalizing this measure to lie in the range [0,1], it becomes quite impartial to K. There are several choices for normalization based on the entropies $H(Y)$ and $H(\hat{Y})$. I shall follow the definition of normalized mutual information (NMI) using geometrical mean, $NMI = \frac{I(Y;\hat{Y})}{\sqrt{H(Y)\cdot H(\hat{Y})}}$, as given in (Strehl and Ghosh, 2002). In practice, one often uses a sample estimate

$$NMI = \frac{\sum_{h,l} n_{h,l} \log\left(\frac{n \cdot n_{h,l}}{n_h n_l}\right)}{\sqrt{\left(\sum_h n_h \log \frac{n_h}{n}\right)\left(\sum_l n_l \log \frac{n_l}{n}\right)}}\ , \tag{2.15}$$

where n_h is the number of data objects in class h, n_l the number of objects in cluster l and $n_{h,l}$ the number of objects in class h as well as in cluster l. The NMI value is 1 when clustering results perfectly match the external category labels and close to 0 for a random partitioning.

In the simplest scenario where the number of clusters equals the number of categories and their one-to-one correspondence can be established, any of these

external measures can be fruitfully applied. For example, when the number of clusters is small (< 4), the accuracy measure is intuitive and easy to understand. However, when the number of clusters differs from the number of original classes, the confusion matrix is hard to read and the accuracy difficult or impossible to calculate. The NMI measure is better than purity and entropy measures, both of which are biased towards high k solutions (Strehl et al., 2000; Strehl and Ghosh, 2002).

Chapter 3

A Unified Framework for Model-based Clustering

In this chapter, I first present a unifying bipartite graph view of probabilistic model-based clustering and demonstrate the benefits of having such a viewpoint. The unifying view provides a good understanding and visualization of existing model-based partitional and hierarchical algorithms. This view also points to several connections between model-based clustering and graph partitioning methods.

In Section 3.2, model-based partitional clustering is analyzed mathematically from a deterministic annealing point of view, which clearly explains the difference and relationship between the generic model-based k-means algorithm and the EM clustering algorithm and establishes the connections among model-based clustering via deterministic annealing, SOM, and Neural-Gas algorithms.

Model-based hierarchical clustering is discussed in Section 3.3, along with several novel inter-cluster distances. A clear distinction between model-based hierarchical clustering and similarity-based hierarchical clustering is made.

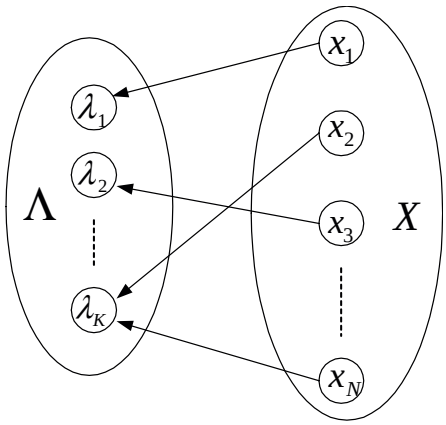


Figure 3.1: A bipartite graph view of model-based clustering.

3.1 A Bipartite Graph View

The bipartite graph view (Fig. 3.1) assumes a set of N data objects, represented by $x_1, x_2, ...,$ and x_N, and K clusters, represented by models $\lambda_1, \lambda_2, ...,$ and λ_K, respectively.[1] In this view, there are data objects X (which can be any complex data types, e.g., sequences) in a data space, probabilistic generative models Λ (e.g., HMMs) in a model space, and connections between the two spaces. Each cluster is represented by a model in the model space, which usually contains models from a specific family of models. The model λ_k can be viewed as the generalized "centroid" of cluster k. A connection between object x_n and model λ_k indicates that the object x_n is being associated with cluster k, with the connection weight (closeness) between them given by the log-likelihood $\log p(x_n|\lambda_k)$.

Several benefits from this bipartite graph view can be immediately observed. Note that this idea of representing clusters by models is radically more general than the standard k-means algorithm, where both data objects and clus-

[1]Recall that I interchangeably use λ to represent a model as well as the set of parameters of that model. Each cluster is described by a model. The set of all parameters used for modeling the whole dataset is represented by $\Lambda = \lambda_1, ..., \lambda_K$. Later in this dissertation, Λ also include mixture weight parameters (α's) for soft clustering.

ter centroids are in the same data space. The models also provide a probabilistic interpretation of clusters, which is a desirable feature in many applications. From this view, one can see that why model-based clustering has a potential to be computationally efficient—there are a total of NK connections to work on. When $K \ll N$, the number NK is much smaller than the number of connections encountered in similarity-based clustering ($N(N-1)/2$).

A variety of hard and soft assignment strategies can be designed by attaching to each connection an association probability based on the connection weights. For hard clustering these probabilities are either 1's or 0's. Intuitively, a suitable objective function is the sum of all connection weights (log-likelihoods) weighted by the association probabilities, which is to be maximized. Indeed, maximizing this objective function leads to a hard clustering algorithm, which I call model-based k-means, as shown in the next section. I will also show in the next section that soft model-based clustering can be obtained by adding entropy constraints to the objective function. Similar to the derivation of deterministic annealing, a temperature parameter can be used to parameterize the softness of association probabilities.

A straightforward design of a model-based partitional clustering algorithm is to iteratively re-train models and re-partition data objects. Partitioning of data objects simply amounts to assigning data to models based on the log-likelihood closeness measures. To train a model λ_k given data objects X_k assigned to it, maximum-likelihood algorithms can be employed to maximize $P(X_k|\lambda_k)$. As shown in the next section, this ML clustering algorithm locally maximizes $P(X|\Lambda)$, or equivalently, $\log P(X|\Lambda)$. Maximum *a priori* (MAP) clustering algorithms can be constructed to locally maximize $P(\Lambda|X)$; they basically amount to using MAP training of models. MAP learning has been shown to be able to smooth the learning process and overcome poor initializations when accurate priors are used (Gauvain and Lee, 1994). But more (prior) parameters need to be specified. In

this dissertation, I focus on ML clustering algorithms but the extension to MAP clustering is fairly straightforward.

One may observe that the above design is just what the EM algorithm does—iteratively computes the (hidden) cluster identities of data objects in the E-step and estimates the model parameters in the M-step. Indeed, for model-based partitional clustering, the bipartite graph shows conceptually what the EM algorithm is about. But the bipartite graph view also provides a good visualization for model-based hierarchical clustering and suggests new variations of existing algorithms. For example, two possible extensions of the model-based partitional clustering based on the bipartite graph view are: (a) to impose a structure on the K models in the model space; (b) to constrain the partitioning of data objects in the data space in certain ways. If a grid map structure is imposed on the relationship between models (Fig. 3.2), one can get a SOM-like model-based partitional clustering algorithm and at the end of clustering process, the relative distance of different clusters should conform to the map structure. For the second extension idea, I will introduce in Chapter 5 a balanced model-based clustering algorithm which can produce balanced clusters and improve clustering quality by using balance constraints in the clustering process.

Now let us set $K = N$ and hierarchically merge clusters in the model space to build a tree structure (Fig. 3.3). This results in the model-based hierarchical clustering algorithm, from a bipartite graph point of view. It is easy to see the difference from the standard single-link or complete-link hierarchical clustering algorithms is that the hierarchy is built in the model space and the inter-cluster distances are calculated from the $KN = N^2$ connection weights between the model space and the data space. In Section 3.3, I will present several empirical inter-cluster distance measures that simplify the computation of model-based hierarchical clustering algorithms.

Before closing this section, I shall point out a few interesting connections

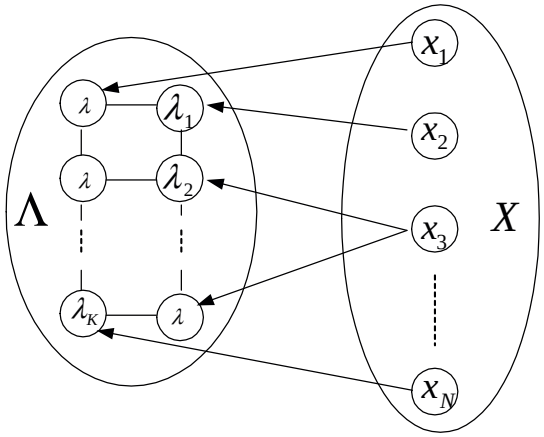


Figure 3.2: Model-based partitional clustering with a grid map structure in the model space.

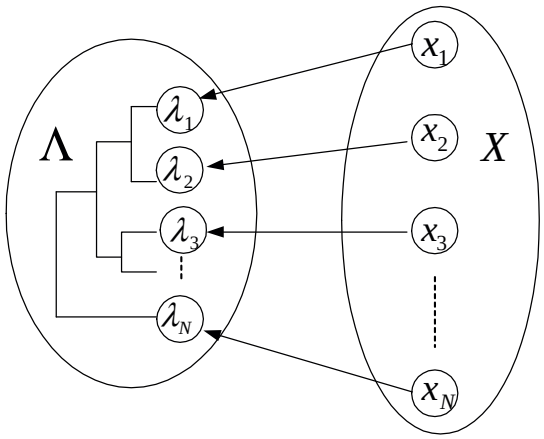


Figure 3.3: A graph view of model-based hierarchical clustering.

between model-based clustering and bipartite graph partitioning. In partitional clustering, hard assignments of data objects to models corresponds to a partitioning of the bipartite graph (Fig. 3.1) with the constraint that each partition contains exactly one model vertex. In fact, the ML data assignment used in the mk-means algorithm in the next section is equivalent to a constrained minimum cut of the bipartite graph into K partitions. For model-based HAC algorithms, merging two clusters corresponds to a partitioning of the graph into $K-1$ clusters in which one partition contains exactly two model vertices and all other partitions have one model vertex each. These connections may point to a way of combining the model-based method with graph partitioning algorithms and deserve more investigation in the future.

3.2 Model-based Partitional Clustering

In this section, I first present three generic model-based partitional clustering algorithms and then discuss how they can be unified from a deterministic annealing point of view. The three generic algorithms are model-based k-means (mk-means), EM clustering, and stochastic mk-means, respectively. Model-based DA clustering algorithms is also analyzed.

3.2.1 Revisiting K-means and EM Clustering

This section revisits the classic k-means and EM clustering algorithms in a generic model-based clustering context. A variation of k-means is also discussed.

Model-based K-means

The model-based k-means (*mk-means*) algorithm (Fig. 3.4) is a generalization of the standard k-means algorithm, with the cluster centroid vectors being replaced

by probabilistic models. It can be easily verified to locally maximize the log-likelihood objective function[2]

$$\log P(X|\Lambda) = \sum_n \log p(x_n|\lambda_{y_n}) \ , \tag{3.1}$$

where $\Lambda = \{\lambda_1, ..., \lambda_k\}$ and $y_n = \arg\max_k \log p(x_n|\lambda_k)$ is the cluster identity of object x_n. The convergence of this generic mk-means algorithm is given by the following theorem.

Theorem 1 *If $p(x|\lambda)$ is bounded from above, the mk-means algorithm given in Fig. 3.4 will converge to a local maximum of the objective function in (3.1).*

Proof: It is easy to verify that both step 2a and 2b in Fig. 3.4 will not decrease the objective function (3.1). The objective function is upper-bounded since $p(o|\lambda)$ is bounded from above. These two conditions complete the convergence proof. □

When equi-variance spherical Gaussian models are used in a vector space, the mk-means is equivalent to the standard k-means algorithm (as discussed in Section 2.3). Many existing model-based clustering methods, which have been explored in a variety of different application domains, can be seen as instances of this generic algorithm. For example, the spherical k-means algorithm (Dhillon and Modha, 2001; Banerjee and Ghosh, 2002a) uses von Mises-Fisher models as its underlying model for text clustering. HMM-based k-means algorithms were explored for sequence clustering (Dermatas and Kokkinakis, 1996; Li and Biswas, 2000). Recently, Bar-Joseph et al. (2002) employed a probabilistic spline model-based clustering method for analyzing gene expression time series.

Mixture-of-Models EM Clustering

The generic EM clustering algorithm (Fig. 3.5) is a generalization of the mixture-of-Gaussians clustering by extending Gaussian models to any probabilistic models

[2]So far I have used $P(x|\lambda)$ for discrete probability mass function and $p(x|\lambda)$ for continuous probability density function. Here I use $p(x|\lambda)$ for the general case where λ could be either a discrete or a continuous model.

Algorithm: mk-means

Input: A set of N data objects $X = \{x_1, ..., x_N\}$, and model structure $\Lambda = \{\lambda_1, ..., \lambda_K\}$.

Output: Trained models Λ and a partition of the data objects given by the cluster identity vector $Y = \{y_1, ...y_N\},\ y_n \in \{1, ..., K\}$.

Steps:

1. Initialization: initialize the model parameters Λ;

2a. Data assignment: for each data object x_n, set

$$y_n = \arg\max_k \log p(x_n|\lambda_k) \ ;$$

2b. Model estimation: for each cluster k, let $X_k = \{x_n|y_n = k\}$, the parameters of each model λ_k is re-estimated as

$$\lambda_k = \arg\max_\lambda \sum_{x \in X_k} \log p(x|\lambda) \ ;$$

3. Stop if Y does not change, otherwise go back to Step 2a.

Figure 3.4: Model-based k-means algorithm.

for which a maximum likelihood estimation is possible (e.g., probabilistic models in the exponential family). It aims at maximizing the data log-likelihood

$$\log P(X|\Lambda) = \sum_n \log \left(\sum_k \alpha_k p(x_n|\lambda_k) \right) , \tag{3.2}$$

assuming that the data objects are generated from the probabilistic models Λ. The algorithm amounts to iterating between the following E-step and M-step until convergence:

E-step:

$$P(k|x_n, \Lambda) = \frac{\alpha_k p(x_n|\lambda_k)}{\sum_j \alpha_j p(x_n|\lambda_j)} \ ; \tag{3.3}$$

M-step:

$$\lambda_k^{(new)} = \arg\max_\lambda \sum_x P(k|x_n, \Lambda) \log p(x|\lambda), \tag{3.4}$$

Algorithm: EM clustering

Input: A set of N data objects $X = \{x_1, ..., x_N\}$, model structure $\Lambda = \{\lambda_1, ..., \lambda_K, \alpha_1, ..., \alpha_K\}$, where λ's are the models and α's are model priors that are subject to $\alpha_k \geq 0, \forall k$ and $\sum_k \alpha_k = 1$.

Output: Trained model parameters Λ that maximizes $\log P(X|\Lambda)$ in (3.2), and a partition of the data objects given by the cluster identity vector $Y = \{y_1, ..., y_N\},\ y_n \in \{1, ..., K\}$.

Steps:

1. Initialization: initialize the model parameters Λ;

2a. E-step: the posterior probability of model k, given a data object x_n and current model parameters Λ, is estimated in (3.3);

2b. M-step: the maximum likelihood estimation of model parameters Λ is given in (3.4) and (3.5).

3. Stop if $\log P(X|\Lambda)$ converges, otherwise go back to Step 2a;

4. For each data object x_n, set $y_n = \arg\max_k \log \alpha_k p(x_n|\lambda_k)$.

Figure 3.5: EM clustering algorithm.

$$\alpha_k^{(new)} = \frac{1}{N} \sum_n P(k|x_n, \Lambda) \ . \tag{3.5}$$

A partition of the data objects is actually a byproduct of the maximum likelihood estimation process.

Special cases of the EM clustering can be obtained by fixing a partial set of parameters at certain values. For example, if the mixture weight parameters are set to be constant ($\alpha_k = 1/K, \forall k$), a *soft mk-means* algorithm is obtained. In this case, the posterior probability in the E-step becomes

$$P(k|x_n, \Lambda) = \frac{p(x_n|\lambda_k)}{\sum_j p(x_n|\lambda_j)} \ . \tag{3.6}$$

Existing instances of the generic EM clustering include the mixture-of-Gaussians clustering (McLachlan and Basford, 1988; Banfield and Raftery, 1993),

the mixture-of-vMFs (Banerjee et al., 2003) and the mixture-of-multinomials (Meila and Heckerman, 2001) for clustering documents, the mixture-of-Markov chains (Cadez et al., 2000) for clustering sequences.

Stochastic Model-based K-means

The *stochastic mk-means* is a stochastic variant of the mk-means. The basic idea is to *stochastically* assign each data object entirely to one cluster (and not fractionally, as in soft clustering), with the probability of object x_n going to cluster k set to be the posterior probability in (3.6). Kearns et al. (1997) described this algorithm as *posterior assignment*. The stochastic mk-means can also be viewed as a sampled version of the EM clustering, where one uses a sampled E-step based on the posterior probability.

3.2.2 A Unified Treatment Via Deterministic Annealing

Let the association probability on each connection weight in the bipartite graph (Fig. 3.1) be $P(x, y)$, where $x \in \{x_1, ..., x_N\}$ and $y \in \{1, ..., K\}$.[3] As mentioned in Section 3.1, the objective function to be maximized can be written as

$$L = \sum_{x,y} P(x, y) \log p(x|\lambda_y) = \sum_{x} P(x) \sum_{y} P(y|x) \log p(x|\lambda_y) \ . \tag{3.7}$$

Comparing this objective with the cost function (2.6) analyzed for deterministic annealing, one can easily see that maximizing (3.7) is equivalent to minimizing the cost function (2.6) with the distortion function $d(x, y)$ set to $-\log p(x|\lambda_y)$. Directly optimizing (3.7) over $P(y|x)$ and Λ leads to the mk-means algorithm shown in Fig. 3.4.

Similar to the derivation of deterministic annealing for clustering, I use an

[3]Here I use y instead of index k for the purpose of an information-theoretic analysis, i.e., to view y as a random variable.

entropy-constrained log-likelihood objective

$$L_1 = L - T \cdot H(Y) + T \cdot H(Y|X) = L - T \cdot I(X;Y) \ . \tag{3.8}$$

Plugging $d(x,y) = -\log p(x|\lambda_y)$ into the optimality conditions (2.10) and (2.11) for deterministic annealing, I get the following solution for maximizing (3.8):

$$P(y|x) = \frac{P(y)p(x|\lambda_y)^{\frac{1}{T}}}{\sum_{y'} P(y')p(x|\lambda_{y'})^{\frac{1}{T}}} \ , \tag{3.9}$$

and

$$\lambda_y^{(new)} = \arg\max_{\lambda} \sum_x P(y|x) \log p(x|\lambda) \ . \tag{3.10}$$

Now I get a model-based partitional clustering algorithm parameterized by a temperature parameter T. The posterior probability $P(y|x)$ in (3.9) is now actually conditioned on current parameters Λ, but for simplicity I use $P(y|x)$ instead of $P(y|x,\Lambda)$ where there is no confusion.

It is easy to see that when $T = 1$, the algorithm reduces to the EM clustering. Plugging (3.9) into L_1 and setting $T = 1$ reduce the objective function to

$$L_1^* = \sum_x P(x) \log \left(\sum_y P(y)p(x|\lambda_y) \right) \ . \tag{3.11}$$

which is exactly the objective function (3.2) that the EM clustering optimizes. As T goes to 0, (3.9) reduces to hard mk-means assignment and the algorithm reduces to mk-means, independent of the actual $P(y)$'s (unless they are 1 and 0's). It can be observed from (3.8) that another way of reducing the EM clustering to mk-means is to first enforce hard assignment ($H(Y|X)$ becomes 0) and then use uniform priors ($H(Y)$ becomes constant), thus reducing the EM objective to L (the mk-means objective) plus some constant. This reduction, however, is not as straightforward as letting $T \to 0$.

Let us get back to the problem of deriving the standard k-means from the mixture-of-Gaussians clustering discussed in Section 2.3. The solution is simple,

one just parameterizes the mixture-of-Gaussians clustering with a temperature parameter T and let T go to zero. Note T simply changes the posterior probability $P(k|x_n, \Lambda)$ but does not alter the probabilistic distribution $p(x|\lambda)$. Interestingly, for the mixture-of-Gaussians clustering the parameter T can be assimilated into the variance parameter σ, resulting in the traditional interpretation that letting the variance of Gaussian distributions go to 0^+ leads to the k-means algorithm.

This analysis makes it clear that mk-means and EM clustering are two special stages of the model-based DA clustering process with $T = 0$ and $T = 1$, respectively, and they optimize two different objective functions (L vs. $L - I(X;Y)$). Since larger T indicates smoother objective function and a smaller number of local solutions, theoretically the EM clustering ($T = 1$) should have a better chance of finding good local solutions than the mk-means algorithm ($T = 0$). It also becomes clear why it makes sense to use the mixture-of-Gaussians clustering results to initialize the standard k-means, which has been heuristically used in practice and can be viewed as a simple one-step deterministic annealing (from $T = 1$ to $T = 0$). Of course, a better approach is to start at a high $T \gg 1$ and gradually reduce T to 0. At each T, the EM algorithm is run until convergence.

Generalizing the SOM and Neural-Gas algorithms to model-based clustering is straightforward: one just substitutes the distortion function $d(x, y)$ by $-\log p(x|\lambda_y)$. This extension points to a way of constructing online model-based DA clustering since both SOM and Neural-Gas algorithms are originally proposed in an online form in the competitive learning context. This also leads to possible extensions of many other existing competitive learning techniques (Xu et al., 1993; Galanopoulos et al., 1997; Zhang and Liu, 2002) to model-based clustering, which are not covered in this dissertation.

A useful observation from (3.9) and (3.10) is that the model-based clustering problem can be decomposed into two subproblems: a data assignment sub-

problem (E-step) and a model estimation subproblem (M-step). It is worth noting that the algorithms in Section 3.2.1 differ in just the E-step. This decomposition was used by Kalton et al. (2001) to generalize the M-step to any supervised learning algorithm. We have used this observation to develop a framework for balanced model-based clustering (Zhong and Ghosh, 2003b) where balance constraints are enforced in the E-step. Such balanced clustering algorithms are useful in several practical applications (see Section 5) and prove to be beneficial to the success of the new hybrid clustering methods proposed in Chapter 6.

3.2.3 Practical Issues and Discussions

In practice, it is common to see the condition $p(x_n|\lambda_{y_n}) \gg p(x_n|\lambda_k), \forall k \neq y_n$ (especially for complex models such as HMMs), which means that $P(k|x_n, \Lambda)$ in (3.3) or (3.6) will be dominated by the likelihood values and be very close to 1 for $k = y_n$, and 0 otherwise, independent of most choices of α's. This suggests that the difference is small between hard mk-means, stochastic mk-means, and soft EM clustering algorithms, i.e., their clustering results will be similar.

The second comment is on the clustering process. In general, the ML estimation of model parameters in (3.4) is itself an iterative optimization process, that needs appropriate initialization (at every iteration of the clustering algorithm) and may get into local minima. For the mk-means and EM clustering algorithms to converge, sequential initialization has to be used. That is, the model parameters from the previous iteration should be used to initialize the ML algorithm for model estimation at the current iteration, to guarantee that the objectives (3.1) and (3.2) do not decrease.

The third observation is that the ML model estimation sometimes leads to a singularity problem, i.e., unbounded log-likelihood. This usually happens for a continuous probability distribution for which $p(x|\lambda)$ is a probability density that

can become very large even though $\int_x p(x|\lambda)dx = 1$. For example, for Gaussian models, this happens when the covariance matrix becomes singular. For discrete distributions, this would not happen since $p(x|\lambda)$ is a probability and upper-bounded by 1. The singularity problem is often dealt in one of the following three ways: (1) restarting the whole clustering algorithm with a different initialization (Juang et al., 1986); (2) using MAP estimation with an appropriate prior (Gauvain and Lee, 1994); (3) using constrained ML estimation, e.g., lower-bound the variance for spherical Gaussian models (Bishop, 1995).

Finally, let us look at the computational complexity for model-based partitional clustering algorithms. Let us first consider partitional clustering and those models for which the estimation of model parameters has a closed-form solution and does not need an iterative process (e.g., Gaussian, Multinomial, etc.). For each iteration, the time complexity is linear in the number of data objects N and the number of clusters K for both the data assignment step and the model estimation step. The total complexity is $O(KNM)$, where M is the number of iterations. For those models for which the estimation of parameters needs an iterative process, the model estimation complexity is $O(KNM_1)$ for each clustering iteration, where M_1 is the number of iterations used in the model estimation process. In this case, the total complexity of the clustering process is $O(KNMM_1)$. Theoretically the number of iterations M and M_1 could be very large but in practice the maximum number of iterations is often set to be a constant. So one can view the complexity of model-based clustering algorithms as linear in K and N in practical applications.

3.3 Model-based Hierarchical Clustering

Researchers usually do not discriminate between model-based and similarity-based approaches for hierarchical clustering algorithms. In contrast, I make a distinction

between model-based hierarchical methods and similarity-based ones. The Ward's algorithm and centroid method are model-based methods. The former algorithm selects two clusters whose merge maximizes the resulting likelihood, whereas the latter algorithm chooses the two clusters whose centroids are closest. Both methods use spherical Gaussian distributions as the underlying models. On the other hand, single-link, complete-link, and average-link methods are all discriminative methods, since data-pairwise distances have to be calculated and form the basis for computing inter-cluster distances.

Kamvar et al. (2002) presented interpretations of several classic hierarchical agglomerative clustering algorithms from a model-based standpoint. They intended to fit HAC algorithms into a standard model-based hierarchical clustering framework and discovered the corresponding model for each of the four agglomerative algorithms (Ward, single-link, complete-link, and average-link). However, only Ward's algorithm fits a natural model interpretation, while the other three match either a contrived model or an approximate one. In my opinion, the Ward algorithm is indeed a model-based approach in that it assumes spherical Gaussian models for each cluster and as a result each cluster can be represented by its mean. The latter three, however, are discriminative algorithms since they are based on data-pairwise similarities or distances. This explains the difficulty of determining suitable underlying generative models for them.

To design model-based hierarchical clustering algorithms, one first needs a methodology for identifying two clusters to merge at each iteration. To do this, I define a distance measure between clusters (i.e., models) and then iteratively merge the closest pair of clusters. A traditional way is to choose the two clusters such that merging them results in the largest log-likelihood $\log P(X|\Lambda)$. The

"distance"[4] for this method can be defined as

$$D^W(\lambda_k, \lambda_j) = \log P(X|\Lambda_{before}) - \log P(X|\Lambda_{after}) \ , \tag{3.12}$$

where Λ_{before} and Λ_{after} are the set of all parameters before and after merging two models (λ_k and λ_j), respectively. I call this measure (generalized) Ward's distance since this is exactly the Ward's algorithm (Ward, 1963) when equi-variance Gaussian models are used. Since merging two clusters and retraining a model for the merged cluster usually lead to a decrease in the log-likelihood, the distance in (3.12) is positive.

This above method is not efficient, however, since to find the closest pair one needs to train a merged model for every pair of clusters and then evaluate the resulting log-likelihood. In practice, except for some specific models for which the Ward's distance can be efficiently computed (Fraley, 1999; Meila and Heckerman, 2001), the Kullback-Leibler (KL) distance measure which does not involve re-estimating models has been commonly used (Sinkkonen and Kaski, 2001; Ramoni et al., 2002). The exact KL divergence is difficult to calculate for complex[5] models; An empirical KL divergence between two models λ_k and λ_j can be defined as

$$D^K(\lambda_k, \lambda_j) = \frac{1}{|X_k|} \sum_{x \in X_k} (\log p(x|\lambda_k) - \log p(x|\lambda_j)) \ , \tag{3.13}$$

where X_k is the set of data objects being grouped into cluster k. This distance can be made symmetric by defining (Juang and Rabiner, 1985)

$$D^K_s(\lambda_k, \lambda_j) = \frac{D^K(\lambda_k, \lambda_j) + D^K(\lambda_j, \lambda_k)}{2} \ ,$$

or using the Jensen-Shannon divergence with $\pi_1 = \pi_2 = \frac{1}{2}$ (Lin, 1991):

$$D^{JS}(\lambda_k, \lambda_j) = \frac{1}{2} D^K(\lambda_k, \frac{\lambda_k + \lambda_j}{2}) + \frac{1}{2} D^K(\lambda_j, \frac{\lambda_k + \lambda_j}{2}) \ .$$

[4]This and several other quantities defined in this section are used as merging criteria and are termed "distance" in a colloquial sense. They may not satisfy the conditions required by being symmetric and obeying triangle inequality.

[5]A complex model has high representational power and is able to describe complex (shaped) data, such as complex-shaped vector data and variable length sequences.

Compared to classical HAC algorithms, KL divergence is analogous to the centroid method. It can be shown that when Gaussian models with equal covariance matrices are used the KL divergence reduces to the Mahalanobis distance between two cluster means. Motivated by this observation as well as the single-link and complete-link HAC algorithms, I propose several new (modified KL) distances. Corresponding to single-link, I define a *minKL* distance as

$$D^m(\lambda_k, \lambda_j) = \min_{x \in X_k} \left(\log p(x|\lambda_k) - \log p(x|\lambda_j)\right) , \tag{3.14}$$

and corresponding to complete-link, I define a *maxKL* distance as

$$D^M(\lambda_k, \lambda_j) = \max_{x \in X_k} \left(\log p(x|\lambda_k) - \log p(x|\lambda_j)\right) . \tag{3.15}$$

Finally, to characterize high "boundary density" between two clusters for building complex-shaped clusters, I propose a *boundaryKL* distance measure

$$D^B(\lambda_k, \lambda_j) = \frac{1}{|B_k|} \sum_{x \in B_k} \left(\log p(x|\lambda_k) - \log p(x|\lambda_j)\right) , \tag{3.16}$$

where B_k is the η fraction of X_k that have smallest $\log p(x|\lambda_k) - \log p(x|\lambda_j)$ values. A value of 0 for $\log p(x|\lambda_k) - \log p(x|\lambda_j)$ defines the "boundary" between cluster k and j. This measure reduces to the minKL distance if B_k contains only one data object, and to the KL distance if $B_k = X_k$. The minKL and maxKL measures are more sensitive to outliers than the KL distance since they are defined on only one specific object. A favorable property of the minKL measure, however, is that hierarchical algorithms using this distance can produce complex-shaped clusters. To guard against outliers but reap the benefits of single-link methods, I set η to be around 10%. The experimental results in Chapter 6 demonstrate the superiority of this new distance measure.

Fig. 3.6 describes a generic model-based HAC algorithm. Instances of this algorithm include existing model-based HAC algorithms that were first explored by Banfield and Raftery (1993) and Fraley (1999) with Gaussian models, later

Algorithm: model-based HAC

Input: A set of N data objects $X = \{x_1, ..., x_N\}$, and model structure λ.

Output: An N-level cluster (model) hierarchy and hierarchical partition of the data objects, with n models/clusters at the n-th level.

Steps:

1. Initialization: start with the N-th level, initialize each data object as itself a cluster and train a model for each cluster, i.e., $\lambda_n = \max_{\lambda} \log p(x_n|\lambda)$;

2a. Distance calculation: compute pairwise inter-cluster distances using an appropriate measure, e.g., one of the measures defined in (3.12)-(3.16);

2b. Cluster merging: merge the two closest clusters (assume they are k and j) and re-estimate a model from the merged data objects $X_k = X_k \cup X_j$, i.e., $\lambda_k = \max_{\lambda} \log P(X_k|\lambda)$;

3. Stop if all data objects have been merged into one cluster, otherwise go back to Step 2a.

Figure 3.6: Model-based hierarchical agglomerative clustering algorithm.

by Vaithyanathan and Dom (2000) with multinomial models for clustering documents, and more recently by Ramoni et al. (2002) with MC models for grouping robot sensor time series. The first three works used the Ward's distance in (3.12) and the last one employed the KL distance in (3.13).

Let us conclude this section with a complexity analysis of model-based hierarchical agglomerative clustering. There are i models/clusters at the i-th level and one starts from N clusters at the bottom. The number of inter-cluster distances to be calculated is $\frac{N(N-1)}{2}$ for the bottom (N-th) level and $i-1$ for the i-th level ($i < N$). The total number of distances calculated for the whole hierarchy is

$$\frac{N(N-1)}{2} + (N-2) + (N-1) + \cdots + 1 \simeq O(N^2) \ .$$

Using the same logic, we can compute the total number of distance comparisons needed as

$$\frac{N(N-1)}{2} + \frac{(N-1)(N-2)}{2} + \cdots + \frac{2 \cdot 1}{2} \simeq O(N^3) \ ,$$

and the total complexity for model estimation as

$$N \cdot 1M_1 + 1 \cdot 2M_1 + 1 \cdot 3M_1 + \cdots + 1 \cdot NM_1 \simeq O(N^2 M_1) \ ,$$

where M_1 is the number of iterations used for model estimation. The complexity can be reduced to $O(N^2 \log N)$ for inter-cluster distance comparisons by using a clever data structure (e.g., heap) to store the comparison results (Jain et al., 1999). Clearly, a complexity of $O(N^3)$ or $O(N^2 \log N)$ is still too high for practical use, which explains why model-base hierarchical clustering algorithms is not as popular as partitional clustering ones. In areas where researchers do use hierarchical algorithms, model-specific tricks have often been used to further reduce the computational complexities (Fraley, 1999; Meila and Heckerman, 2001).

Chapter 4

Document Clustering: A Case Study

This chapter presents a comparative case study on document clustering based on the model-based clustering framework proposed in Chapter 3. This study shows that different existing models can be readily plugged into the framework for a fair and useful comparison. In particular, I compare three generative models for text documents: multivariate Bernoulli, multinomial, and von Mises-Fisher distributions. Four model-based clustering algorithms—mk-means, stochastic mk-means, EM clustering, and DA clustering—are studied. The first three represent three different data assignment strategies and the last one anneals the EM assignment. The experimental results over a large number of datasets show that, in terms of clustering quality, (a) The Bernoulli model performs the worst and is inadequate for text clustering; (b) The vMF model produces better clustering results than both Bernoulli and multinomial models; (c) Soft assignment (EM) leads to slightly better results than hard assignment (for mk-means & stochastic mk-means); (d) The DA-based algorithms improves EM clustering significantly on some datasets. All these model-based algorithms are also compared with a state-of-the-art discriminative approach to document clustering based on graph partitioning (CLUTO) and a spectral co-clustering method. Overall, CLUTO and

DA clustering perform the best but are also the most computationally expensive; the spectral co-clustering algorithm fares worse than the vMF-based methods.

The organization of this chapter is as follows. Section 4.1 gives the motivation for this case study. Section 4.2 describes the three probabilistic models for clustering text documents. Section 4.3–4.5 introduce a number of document datasets, experimental settings, and compare the performance of different models and data assignment strategies. Finally, section 4.6 concludes this chapter.

4.1 Motivation

Document clustering has become an increasingly important technique for unsupervised document organization, automatic topic extraction, and fast information retrieval or filtering. For example, a web search engine often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information. Clustering methods can be used to automatically group the retrieved documents into a list of meaningful categories, as is achieved by search engines such as Northern Light and Vivisimo. Similarly, a large database of documents can be pre-clustered to facilitate query processing by searching only the cluster that is closest to the query.

Till the mid-nineties, hierarchical agglomerative clustering using a suitable similarity measure such as cosine, Dice or Jaccard, formed the dominant paradigm for clustering documents (Rasmussen, 1992; Cutting et al., 1992). The increasing interest in processing larger collections of documents has led to a new emphasis on designing more efficient and effective techniques, leading to an explosion of diverse approaches to the document clustering problem, including (multi-level) self-organizing map (Kohonen et al., 2000), mixture-of-Gaussians (Tantrum et al., 2002), spherical k-means (Dhillon and Modha, 2001), bi-secting k-means (Steinbach et al., 2000), mixture-of-multinomials (Vaithyanathan and Dom, 2000;

Meila and Heckerman, 2001), multi-level graph partitioning (Karypis, 2002), and co-clustering using bipartite spectral graph partitioning (Dhillon, 2001).

McCallum and Nigam (1998) performed a comparative study of Bernoulli and multinomial models for text classification but not for clustering. Comparisons of different document clustering methods have been done by Steinbach, Karypis, and Kumar (2000), and by Zhao and Karypis (2001). They both focused on comparing partitional with hierarchical approaches either for one model, or for similarity-based clustering algorithms (in the CLUTO toolkit). Meila and Heckerman (2001) have compared hard with soft assignment strategies for text clustering using multinomial models. To the best of my knowledge, however, a comprehensive comparison of different probabilistic models for clustering documents has not been done before. A central goal of this chapter is to fill this void. Also I aim to empirically investigate the suitability of each model for document clustering and identify which model works better in what situations.

4.2 Probabilistic Models for Text Documents

The three models used in my experiments, multivariate Bernoulli, multinomial, and vMF, have been introduced in Section 2.2. Here I briefly summarize a few things specific to my experiments.

Data Representation

For multinomial models, the traditional vector space representation is used for text documents, i.e., each document is represented as a high dimensional vector of "word"[1] counts in the document. The dimensionality equals the number of words in the vocabulary used.

[1]Used in a broad sense since it may represent individual words, stemmed words, tokenized words, or short phrases.

Binarized document vectors are used for multivariate Bernoulli models. The binarization obviously loses important word frequency information in each document. Although this representation fares well for text classification (McCallum and Nigam, 1998), it leads to poor clustering results, as shown in Section 4.5.

For vMF models, the word-count document vectors are log(IDF)-weighted and then L_2-normalized. The IDF stands for *inverse document frequency*. The log(IDF) weighting is a common practice in the information retrieval community to de-emphasize the words that appear in too many documents. The L_2 normalization is required since the vMF distribution is a directional distribution defined on a unit hypersphere and does not capture any magnitude information.

Parameter Estimation

To avoid zero probability or singularity problem, MAP estimation is used for learning the parameters of multivariate Bernoulli and multinomial models. Let $P_k^{(l)}$ be the l-th dimension of the probability parameter for the k-th cluster. Employing a Laplacian prior, one can derive the parameter estimation formula for multivariate Bernoulli as (McCallum and Nigam, 1998)

$$P_k^{(l)} = \frac{1 + \sum_x P(k|x, \Lambda) x^{(l)}}{2 + \sum_x P(k|x, \Lambda)} \ , \tag{4.1}$$

and the formula for multinomial as

$$P_k^{(l)} = \frac{1 + \sum_x P(k|x, \Lambda) x^{(l)}}{\sum_l (1 + \sum_x P(k|x, \Lambda) x^{(l)})} = \frac{1 + \sum_x P(k|x, \Lambda) x^{(l)}}{|V| + \sum_l \sum_x P(k|x, \Lambda) x^{(l)}} \ , \tag{4.2}$$

where $|V|$ is the size of the word vocabulary, i.e., the dimensionality of document vectors, and $P(k|x, \Lambda)$ is the posterior probability of cluster k. Note the parameter $P_k^{(l)}$ has different meaning for multivariate Bernoulli and multinomial models. For Bernoulli models, it is the probability of the l-th word being present in cluster k. For multinomial models, the $P_k^{(l)}$'s describe the word distribution in cluster k and are subject to $\sum_l P_k^{(l)} = 1$.

The estimation of μ in vMF models is straightforward (see Section 2.2). The estimation for κ in the mixture-of-vMFs clustering algorithm, however, is rather difficult due to the Bessel function involved. In (Banerjee et al., 2003), the EM based maximum likelihood solution has been derived, including updates for κ. Even using an approximation for estimating κ's, however, it is computationally much more expensive than the vMF-based k-means algorithm. In this work, for convenience, I use a simpler soft assignment scheme that is similar to deterministic annealing. I use a κ that is constant across all models at each iteration, start with a low value of κ, and gradually increase the κ (i.e. make the distributions more peaked) in unison with each iteration. Note that κ has the effect of an "inverse temperature" parameter in the training process.

4.3 Text Datasets

I used the 20-newsgroups data[2] and a number of datasets from the CLUTO toolkit[3] (Karypis, 2002). These datasets provide a good representation of different characteristics: the number of documents ranges from 204 to 19949, the number of terms from 5832 to 43586, the number of classes from 3 to 20, and the balance from 0.036 to 0.998. Here the balance of a dataset is defined as the ratio of the number of documents in the smallest class to the number of documents in the largest class. So a value close to 1(0) indicates a very (un)balanced dataset. A summary of all the datasets used in this chapter is shown in Table 4.1.

The *NG20* dataset is a collection of 20,000 messages, collected from 20 different usenet newsgroups, 1,000 messages from each. I preprocessed the raw dataset using the Bow toolkit (McCallum, 1996), including chopping off headers and removing stop words as well as words that occur in less than three documents.

[2]http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.
[3]http://www.cs.umn.edu/~karypis/CLUTO/files/datasets.tar.gz.

In the resulting dataset, each document is represented by a 43,586-dimensional sparse vector and there are a total of 19,949 documents (empty documents are removed). The *NG17-19* dataset is a subset of NG20, containing ~ 1000 messages from each of the three categories on different aspects of politics. These three categories are expected to be difficult to separate. After the same preprocessing step, the resulting dataset consists of 2,998 documents in a 15,810 dimensional vector space.

Table 4.1: Summary of text datasets. (For each dataset, n_d is the total number of documents, n_w the total number of words, k the number of classes, and $\bar{n}_c$ the average number of documents per class.)

Data	Source	n_d	n_w	k	$\bar{n}_c$	Balance
NG20	20 Newsgroups	19949	43586	20	997	0.991
NG17-19	3 overlapping subgroups from NG20	2998	15810	3	999	0.998
classic	CACM/CISI/ CRANFIELD/MEDLINE	7094	41681	4	1774	0.323
ohscal	OHSUMED-233445	11162	11465	10	1116	0.437
k1b	WebACE	2340	21839	6	390	0.043
hitech	San Jose Mercury (TREC)	2301	10080	6	384	0.192
reviews	San Jose Mercury (TREC)	4069	18483	5	814	0.098
sports	San Jose Mercury (TREC)	8580	14870	7	1226	0.036
la1	LA Times (TREC)	3204	31472	6	534	0.290
la12	LA Times (TREC)	6279	31472	6	1047	0.282
la2	LA Times (TREC)	3075	31472	6	513	0.274
tr11	TREC	414	6429	9	46	0.046
tr23	TREC	204	5832	6	34	0.066
tr41	TREC	878	7454	10	88	0.037
tr45	TREC	690	8261	10	69	0.088

All the datasets associated with the CLUTO toolkit have already been preprocessed (Zhao and Karypis, 2001) and I further removed those words that appear in two or fewer documents. The *classic* dataset was obtained by combining the CACM, CISI, CRANFIELD, and MEDLINE abstracts that were used in

the past to evaluate various information retrieval systems[4]. CACM consists of 3,203 abstracts from computer systems papers, CISI consists of 1,460 abstracts from information retrieval papers, MEDLINE consists of 1,033 abstracts from medical journals, and CRANFIELD consists of 1,398 abstracts from aeronautical systems papers. The *ohscal* dataset is from the OHSUMED collection (Hersh et al., 1994). It contains 11,162 documents from the following ten categories: antibodies, carcinoma, DNA, in-vitro, molecular sequence data, pregnancy, prognosis, receptors, risk factors, and tomography. The *k1b* dataset is from the WebACE project (Han et al., 1998). Each document corresponds to a web page listed in the subject hierarchy of Yahoo! (http://www.yahoo.com). The other datasets are from TREC collections (http://trec.nist.gov). In particular, the *hitech*, *reviews*, and *sports* were derived from the San Jose Mercury newspaper articles. The *hitech* dataset contains documents about computers, electronics, health, medical, research, and technology; the *reviews* dataset contains documents about food, movies, music, radio, and restaurants; the *sports* dataset contains articles about baseball, basketball, bicycling, boxing, football, golfing, and hockey. The *la1*, *la12*, and *la2* datasets were obtained from articles of the Los Angeles Times in the following six categories: entertainment, financial, foreign, metro, national, and sports. Datasets *tr11*, *tr23*, *tr41*, and *tr45* were derived from TREC-5, TREC-6, and TREC-7 collections.

4.4 Experimental Setting

First I shall compare three probabilistic models—multivariate Bernoulli, multinomial, and von Mises-Fisher, for three types of data assignments each, leading to a total of nine algorithms. All the three models directly handle high dimensional vectors without dimensionality reduction, and have been recommended for doc-

[4]Available from ftp://ftp.cs.cornell.edu/pub/smart.

ument clustering, which involves grouping of vectors that are high-dimensional, sparse with only non-negative entries, and directional (i.e., only the vectors' directions are important as they are typically normalized to unit length). In contrast, Gaussian based models such as k-means perform very poorly for such datasets (Strehl et al., 2000). All nine instantiated algorithms are compared on a number of document datasets derived from the TREC collections and internet newsgroups. I also used deterministic annealing (DA) as a more sophisticated soft clustering approach on multinomial and vMF models[5] and compared all the model-based algorithms with the state-of-the-art graph-based approaches, the CLUTO (Karypis, 2002) algorithm and a bipartite spectral method.

For simplicity, the three algorithms based on the Bernoulli model are named k-Bernoullis, stochastic k-Bernoullis, and mixture-of-Bernoullis, abbreviated as *kberns*, *skberns*, and *mixberns*, respectively. Similarly, the abbreviated names are *kmns*, *skmns*, and *mixmns* for multinomial-based algorithms, and are *kvmfs*, *skvmfs*, and *softvmfs* for vMF-based algorithms. I use *softvmfs* instead of *mixvmfs* for the soft vMF-based algorithm for the following reason. As mentioned in Section 4.2, the estimation of parameter κ in a vMF model is difficult but is needed for the mixture-of-vMFs algorithm. As a simple heuristic, I use $\kappa(m) = 20m$, where m is the iteration number. So κ is a constant for all clusters at each iteration, and gradually increases over iterations. Realizing that this is purely ad-hoc, I also implemented the standard deterministic annealing for the *softvmfs*: (a) a constant κ is used for all clusters at each iteration; (b) the algorithm runs until convergence for each κ; (c) κ follows an exponential schedule $\kappa(m+1) = 1.1\kappa(m)$, starting from 1 and up to 500. I call this algorithm *davmfs*.

For the multinomial-based DA clustering, which I call *damns*, an inverse temperature parameter $\beta = 1/T$ is used to parameterize the E-step of *mixmns*.

[5]Bernoulli models perform very poorly and so do Bernoulli-based DA clustering algorithms. Therefore I do not show the results on the latter algorithms.

The annealing schedule for β is set to $\beta(m+1) = 1.3\beta(m)$, and β starts from 0.5 and can go up to 200. An implementation detail that is worth mentioning is that I used a log-likelihood normalized by the document length, i.e.,

$$\log \tilde{P}(x|\lambda) = \frac{1}{\sum_l x^{(l)}} \log P(x|\lambda) \ ,$$

which results in a more stable annealing process in my experiments.

For all the model-based algorithms (except for *damns* and *davmfs*), I use a maximum number of iterations of 20 (to make a fair comparison). Each experiment is run ten times, each time starting from a different random initialization. The averages and standard deviations of the NMI (2.15) and running time results are reported.

Two state-of-the-art graph-based clustering algorithms are also included in the experiments. The first one is CLUTO (Karypis, 2002), a clustering toolkit based on the Metis graph partitioning algorithms (Karypis and Kumar, 1998). It is worth mentioning that CLUTO is positioned for clustering and drops the strong balance constraints in the original Metis algorithm. I use *vcluster* in the toolkit with the default setting. The other one is a modification of the bipartite spectral co-clustering algorithm (Dhillon, 2001). The modification is according to (Ng et al., 2002)[6] and generates slightly better results than the original bipartite clustering algorithm. Both graph partitioning algorithms uses fast heuristics and thus is dependent on the order of nodes from the input graph. I run each algorithm ten times, each run using a different order of documents.

4.5 Experimental Results and Discussions

Table 4.2, 4.3, 4.4, 4.5, and 4.6 show the NMI results on the *NG20*, *NG17-19*, *classic*, *ohscal*, and *hitech* datasets, respectively, across different number of clusters

[6]Use k instead of $\log k$ eigen-directions and normalize each projected data vector.

for each dataset. All numbers in the table are shown in the format *average* ± *1 standard deviation*. Boldface entries highlight the best algorithms in each column. The number of clusters K does not seem to affect much the relative comparison between different algorithms (at least for the range of K I have experimented with in this study). This is also the case for other datasets. Therefore, to save space, I show the NMI results on all other datasets for one specific K only in Table 4.7 and 4.8.

Table 4.2: NMI Results on *NG20* dataset

K	10	**20**	30	40
kberns	.18 ± .03	.20 ± .04	.18 ± .03	.18 ± .02
skberns	.19 ± .04	.21 ± .03	.19 ± .02	.20 ± .03
mixberns	.18 ± .05	.19 ± .03	.17 ± .02	.18 ± .03
kmns	.50 ± .02	.53 ± .03	.53 ± .02	.54 ± .02
skmns	.51 ± .02	.53 ± .03	.54 ± .02	.55 ± .02
mixmns	.52 ± .02	.54 ± .03	.54 ± .02	.56 ± .02
kvmfs	.53 ± .02	.55 ± .02	.52 ± .01	.50 ± .01
skvmfs	.54 ± .01	.56 ± .01	.48 ± .16	.52 ± .01
softvmfs	.55 ± .02	.57 ± .02	.56 ± .01	.55 ± .01
damns	.55 ± .03	.57 ± .02	.55 ± .02	.53 ± .01
davmfs	**.57 ± .03**	**.59 ± .02**	.57 ± .01	.56 ± .01
CLUTO	.55 ± .02	.58 ± .01	**.58 ± .01**	**.57 ± .01**
co-cluster	.36 ± .01	.46 ± .01	.50 ± .01	.51 ± .01

Of the three models, the vMF model performs the best and the multivariate Bernoulli model the worst. The Bernoulli-based algorithms significantly underperform the other methods for all the datasets except for *ohscal*. This indicates that noting only whether or not a word occurs in a document, but not the number of occurrences, is a limited representation. The vMF-based algorithms perform better than the multinomial-based ones, especially for most of the smaller datasets, i.e., *NG17-19*, *k1b*, *hitech*, *tr11*, *tr23*, *tr41*, and *tr45*. The deterministic annealing algorithm improves the performance of corresponding soft clustering

Table 4.3: NMI Results on *NG17-19* dataset

K	**3**	5	7	9
kberns	$.03 \pm .01$	$.09 \pm .05$	$.08 \pm .03$	$.09 \pm .05$
skberns	$.03 \pm .01$	$.08 \pm .05$	$.09 \pm .04$	$.09 \pm .05$
mixberns	$.03 \pm .01$	$.08 \pm .04$	$.08 \pm .04$	$.08 \pm .05$
kmns	$.23 \pm .08$	$.26 \pm .05$	$.23 \pm .04$	$.23 \pm .04$
skmns	$.22 \pm .08$	$.26 \pm .06$	$.24 \pm .05$	$.23 \pm .04$
mixmns	$.23 \pm .08$	$.27 \pm .05$	$.25 \pm .04$	$.25 \pm .04$
kvmfs	$.37 \pm .10$	$.37 \pm .02$	$.33 \pm .03$	$.32 \pm .03$
skvmfs	$.37 \pm .08$	$.37 \pm .05$	$.38 \pm .03$	$.35 \pm .03$
softvmfs	$.39 \pm .10$	$.40 \pm .04$	$.39 \pm .04$	$.37 \pm .02$
damns	$.36 \pm .12$	$.37 \pm .06$	$.36 \pm .02$	$.36 \pm .02$
davmfs	$\mathbf{.46 \pm .01}$	$\mathbf{.40 \pm .02}$	$.41 \pm .03$	$.39 \pm .02$
CLUTO	$\mathbf{.46 \pm .01}$	$\mathbf{.40 \pm .01}$	$\mathbf{.45 \pm .01}$	$\mathbf{.43 \pm .01}$
co-cluster	$.02 \pm .01$	$.16 \pm .07$	$.36 \pm .03$	$.37 \pm .01$

Table 4.4: NMI Results on *classic* dataset

K	**4**	6	8	10
kberns	$.23 \pm .10$	$.25 \pm .11$	$.25 \pm .08$	$.26 \pm .07$
skberns	$.23 \pm .11$	$.22 \pm .13$	$.21 \pm .10$	$.27 \pm .16$
mixberns	$.20 \pm .15$	$.18 \pm .15$	$.18 \pm .12$	$.18 \pm .17$
kmns	$.56 \pm .06$	$.58 \pm .03$	$.58 \pm .03$	$.58 \pm .02$
skmns	$.57 \pm .06$	$.58 \pm .03$	$.58 \pm .02$	$.56 \pm .02$
mixmns	$.66 \pm .04$	$\mathbf{.65 \pm .04}$	$\mathbf{.64 \pm .03}$	$\mathbf{.65 \pm .02}$
kvmfs	$.54 \pm .03$	$.60 \pm .04$	$.57 \pm .05$	$.56 \pm .04$
skvmfs	$.54 \pm .02$	$.63 \pm .04$	$.61 \pm .03$	$.57 \pm .03$
softvmfs	$.55 \pm .03$	$.61 \pm .06$	$.60 \pm .03$	$.58 \pm .02$
damns	$\mathbf{.71 \pm .06}$	$.63 \pm .06$	$.59 \pm .06$	$.60 \pm .04$
davmfs	$.51 \pm .01$	$.62 \pm .01$	$.60 \pm .01$	$.59 \pm .01$
CLUTO	$.54 \pm .02$	$.64 \pm .01$	$.60 \pm .01$	$.58 \pm .01$
co-cluster	$.01 \pm .01$	$.43 \pm .02$	$.43 \pm .02$	$.59 \pm .03$

Table 4.5: NMI Results on *ohscal* dataset

K	5	**10**	15	20
kberns	.37 ± .02	.37 ± .02	.38 ± .02	.38 ± .03
skberns	.38 ± .01	.38 ± .02	.39 ± .02	.39 ± .03
mixberns	.38 ± .01	.37 ± .02	.38 ± .02	.38 ± .03
kmns	.37 ± .01	.37 ± .02	.37 ± .01	.36 ± .01
skmns	.37 ± .01	.37 ± .02	.37 ± .02	.37 ± .01
mixmns	.37 ± .01	.37 ± .02	.38 ± .02	.37 ± .01
kvmfs	.40 ± .03	.43 ± .03	.41 ± .01	.39 ± .01
skvmfs	.39 ± .02	.44 ± .02	.41 ± .01	.38 ± .01
softvmfs	.40 ± .02	.44 ± .02	.41 ± .01	.41 ± .01
damns	.38 ± .01	.39 ± .02	.39 ± .02	.39 ± .02
davmfs	.41 ± .01	**.47 ± .02**	**.45 ± .01**	.42 ± .01
CLUTO	**.44 ± .01**	.44 ± .02	.44 ± .01	**.43 ± .01**
co-cluster	.39 ± .01	.39 ± .01	.36 ± .01	.38 ± .01

Table 4.6: NMI Results on *hitech* dataset

K	4	**6**	8	10
kberns	.12 ± .05	.11 ± .05	.11 ± .03	.10 ± .04
skberns	.12 ± .02	.12 ± .03	.11 ± .03	.10 ± .04
mixberns	.12 ± .02	.11 ± .04	.11 ± .03	.09 ± .04
kmns	.25 ± .03	.23 ± .03	.23 ± .01	.22 ± .02
skmns	.26 ± .04	.23 ± .04	.23 ± .02	.21 ± .02
mixmns	.26 ± .03	.23 ± .03	.23 ± .02	.23 ± .02
kvmfs	.29 ± .02	.28 ± .02	.27 ± .01	.27 ± .02
skvmfs	.29 ± .02	.29 ± .02	.28 ± .02	.28 ± .02
softvmfs	.30 ± .01	.29 ± .01	.29 ± .02	.30 ± .01
damns	.25 ± .03	.27 ± .01	.29 ± .01	.30 ± .01
davmfs	.32 ± .01	.30 ± .01	.30 ± .02	.30 ± .02
CLUTO	**.34 ± .01**	**.33 ± .01**	**.32 ± .01**	**.32 ± .01**
co-cluster	.26 ± .03	.22 ± .03	.23 ± .01	.24 ± .02

Table 4.7: NMI Results on *hitech*, *reviews*, *sports*, *la1*, *la12*, and *la2* datasets

	reviews	*sports*	*la1*	*la12*	*la2*
K	5	7	6	6	6
kberns	.30 ± .05	.39 ± .06	.04 ± .04	.06 ± .06	.17 ± .03
skberns	.30 ± .04	.37 ± .05	.06 ± .05	.07 ± .06	.19 ± .03
mixberns	.29 ± .05	.37 ± .05	.05 ± .05	.06 ± .05	.20 ± .04
kmns	.55 ± .08	.59 ± .06	.39 ± .05	.42 ± .04	.47 ± .04
skmns	.55 ± .08	.58 ± .06	.41 ± .05	.43 ± .04	.47 ± .05
mixmns	**.56 ± .08**	.59 ± .06	.41 ± .05	.43 ± .05	.48 ± .04
kvmfs	.53 ± .06	.57 ± .08	.49 ± .05	.50 ± .03	.54 ± .04
skvmfs	.53 ± .07	.61 ± .04	.51 ± .04	.51 ± .04	.52 ± .03
softvmfs	**.56 ± .06**	.60 ± .05	.52 ± .04	.53 ± .05	.49 ± .04
damns	.51 ± .06	.57 ± .04	.49 ± .02	.54 ± .03	.45 ± .03
davmfs	**.56 ± .09**	.62 ± .05	.53 ± .03	.52 ± .02	.52 ± .04
CLUTO	.52 ± .01	**.67 ± .01**	**.58 ± .02**	**.56 ± .01**	**.56 ± .01**
co-cluster	.40 ± .07	.56 ± .02	.41 ± .05	.42 ± .07	.41 ± .02

Table 4.8: NMI Results on *k1b*, *tr11*, *tr23*, *tr41*, and *tr45* datasets

	k1b	*tr11*	*tr23*	*tr41*	*tr45*
K	6	9	6	10	10
kberns	.32 ± .25	.07 ± .02	.11 ± .01	.27 ± .05	.13 ± .06
skberns	.36 ± .24	.08 ± .02	.11 ± .01	.27 ± .06	.13 ± .05
mixberns	.31 ± .24	.07 ± .02	.11 ± .01	.27 ± .04	.13 ± .06
kmns	.55 ± .04	.39 ± .07	.15 ± .03	.49 ± .03	.43 ± .05
skmns	.55 ± .05	.39 ± .08	.15 ± .02	.50 ± .04	.43 ± .05
mixmns	.56 ± .04	.39 ± .07	.15 ± .03	.50 ± .03	.43 ± .05
kvmfs	.60 ± .03	.52 ± .03	.33 ± .05	.59 ± .03	.65 ± .03
skvmfs	.60 ± .02	.57 ± .04	.34 ± .05	.62 ± .03	.65 ± .05
softvmfs	.60 ± .04	.60 ± .05	.36 ± .04	.62 ± .05	.66 ± .03
damns	.61 ± .04	.61 ± .02	.31 ± .03	.61 ± .05	.56 ± .03
davmfs	**.67 ± .04**	.66 ± .04	.41 ± .03	**.69 ± .02**	**.68 ± .05**
CLUTO	.62 ± .03	**.68 ± .02**	**.43 ± .02**	.67 ± .01	.62 ± .01
co-cluster	.60 ± .01	.53 ± .03	.22 ± .01	.51 ± .02	.50 ± .03

algorithms, sometimes significantly. Table 4.9 shows the performance gains of *damns* over *mixmns* and *davmfs* over *softvmfs*, on a sorted list of the datasets according to data sizes. A trend seen is that the DA clustering algorithms gain more on medium to small ($n_d \leq 3,000$) datasets.

Table 4.9: Summary of results. (For each dataset, n_d is the total number of documents, $Gain_{mns} = \frac{NMI_{damns} - NMI_{mixmns}}{NMI_{mixmns}}$ the performance improvement of *damns* over *mixmns*, and $Gain_{vmfs} = \frac{NMI_{davmfs} - NMI_{softvmfs}}{NMI_{softvmfs}}$ the performance improvement of *davmfs* over *softvmfs*.)

Data	n_d	Best three algorithms	$Gain_{mns}$	$Gain_{vmfs}$
NG20	19949	davmfs, CLUTO, damns	5.6%	3.5%
ohscal	11162	davmfs, CLUTO, softvmfs	5.4%	6.8%
sports	8580	CLUTO, davmfs, softvmfs	-3.4%	3.3%
classic	7094	damns, mixmns, skmns	7.8%	-7.3%
la12	6279	CLUTO, damns, softvmfs	25.6%	1.2%
reviews	4069	davmfs, softvmfs, mixmns	-8.9%	0%
la1	3204	CLUTO, davmfs, softvmfs	19.5%	1.9%
la2	3075	CLUTO, kvmfs, skvmfs	-6.3%	6.1%
NG17-19	2998	davmfs, CLUTO, softvmfs	56.5%	17.9%
k1b	2340	davmfs, CLUTO, damns	8.9%	11.7%
hitech	2301	CLUTO, davmfs, softvmfs	60.9%	3.3%
tr41	878	davmfs, CLUTO, skvmfs	22.0%	11.3%
tr45	690	davmfs, softvmfs, kvmfs	30.2%	3.0%
tr11	414	CLUTO, davmfs, damns	56.4%	10.0%
tr23	204	CLUTO, davmfs, softvmfs	106.7%	13.9%

To see the annealing effect of the *damns* and *davmfs* approaches, I draw the training curves for both the average (normalized) log-likelihood objective and the average normalized entropy of posterior distribution, in Fig 4.1 and 4.2. The average normalized posterior entropy is calculated by

$$\bar{H}_{post} = \frac{1}{N} \sum_x H(P(y|x, \Lambda)) / \log K \ .$$

It can be seen that at some stage of the annealing process, the average log-likelihood jumps and the entropy $\bar{H}_{post}$ drops quickly to close to zero. This stage

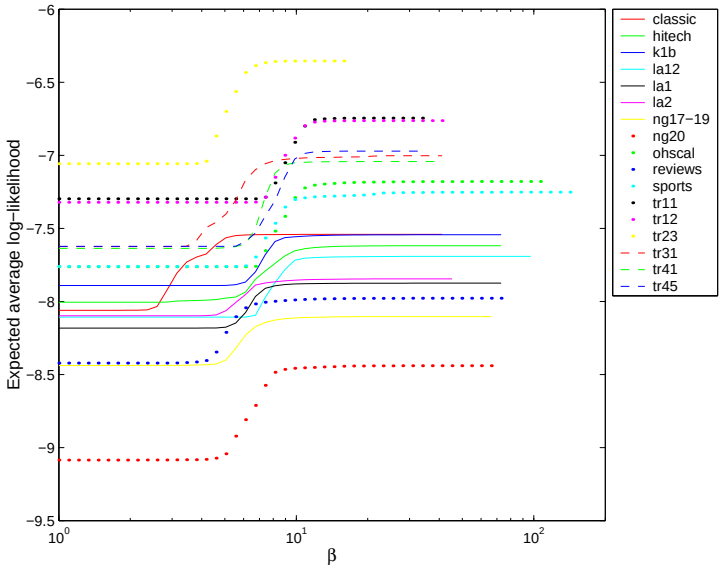
corresponds to the phase transition period in an annealing process. Also it can be seen the phase transitions for different datasets occur at different stages of the annealing process. To achieve good training results, one has to use a temperature schedule that is slow enough not to skip the important phase transition periods (Rose, 1998).

The three different data assignment strategies produce very comparable clustering results across all datasets. The soft assignment is only slightly better than the other two (except for the *classic* dataset where the soft assignment with the multinomial model is clearly the best). For the vMF models, however, the exact EM clustering (Banerjee et al., 2003) can achieve significant improvement over hard assignment.

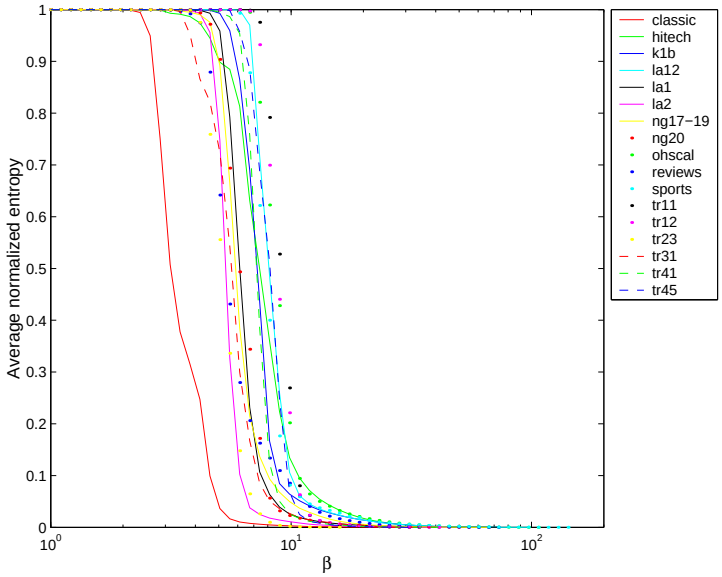
Surprisingly, the bipartite spectral co-clustering algorithm mostly underperforms the vMF-based methods and sometimes gives very poor results (with NMI values close to 0). The other graph-based algorithm, CLUTO, performs much better and is overall one of the best among all the algorithms I have compared. This is not surprising since it is built on a very sophisticated multi-level graph partitioning engine (Karypis and Kumar, 1998). The disadvantage of this approach, and similarity-based algorithms in general, lies in its $O(N^2)$ computational complexity.

Note that the standard deviations of the model-based clustering results are much larger than that of the CLUTO results, indicating that the initialization effect of model-based methods is large. It also means that if one can develop a good initialization strategy to make the results of model-based clustering lean towards the upper end of the performance range, results comparable to the CLUTO results can be obtained. Deterministic annealing improves the local solutions but still sees moderate variation over 10 runs. How to substantially improve the initialization or robustness of model-based clustering remains a challenging problem.

Table 4.10 shows the running time results on *NG20*, the largest dataset

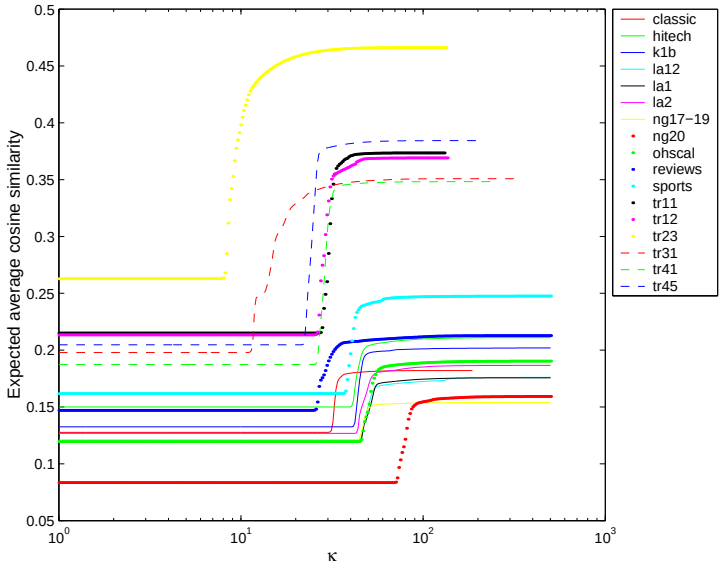


(a)

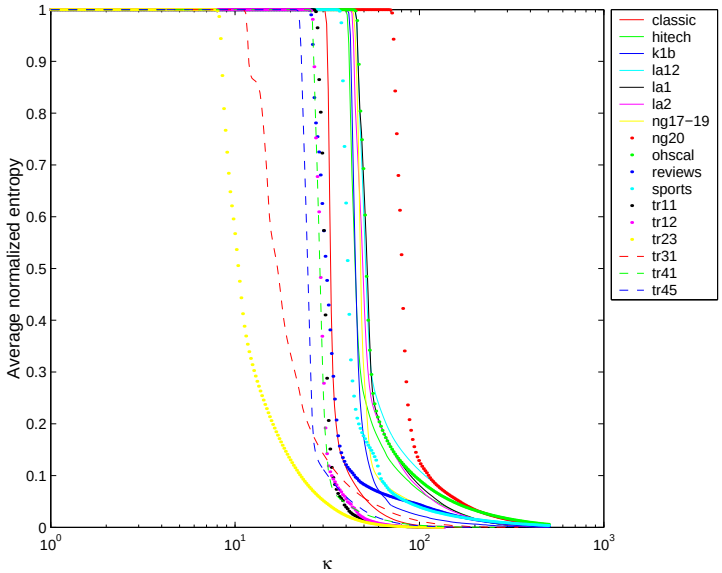


(b)

Figure 4.1: Training curves of multinomial model-based DA clustering: (a) average log-likelihood (normalized by document length); (b) average entropy of posterior distributions. The parameter β is an inverse "temperature" parameter.



(a)



(b)

Figure 4.2: Training curves of vMF model-based DA clustering: (a) average cosine similarity; (b) average entropy of posterior distributions. The temperature parameter is assimilated into the dispersion parameter κ.

used in the experiments. All the numbers are recorded on a 2.4GHz PC running Windows 2000 with memory large enough to hold an individual dataset, and reflect only the clustering time, not including the data I/O cost. Clearly, algorithms using soft assignment take longer time than those using hard assignments, suggesting that one could choose the hard versions in practice when the soft version does not buy much performance (this seems to be the case for Bernoulli and multinomial models according to the NMI results presented above). Overall, the kvmfs algorithm is the fastest one. Since the CLUTO software package is written in C but all the other algorithms are in Matlab, I expect that the first nine model-based algorithms, if re-written in C, will be considerably faster than CLUTO.

Table 4.10: Running time Results on *NG20* dataset (in seconds)

	NG20			
K	10	20	30	40
kberns	26.8 ± 10.6	43.0 ± 19.0	81.6 ± 37.6	125.4 ± 43.6
skberns	30.2 ± 9.8	65.9 ± 22.1	92.3 ± 35.2	144.7 ± 51.8
mixberns	28.5 ± 11.4	77.8 ± 25.4	102.0 ± 38.9	164.9 ± 38.9
kmns	17.5 ± 2.9	36.7 ± 4.9	54.8 ± 7.0	78.5 ± 8.4
skmns	19.7 ± 3.0	39.1 ± 5.6	68.4 ± 7.0	94.9 ± 9.9
mixmns	23.8 ± 3.6	47.7 ± 6.8	74.2 ± 10.0	99.5 ± 12.7
kvmfs	11.4 ± 1.3	17.5 ± 0.3	21.7 ± 0.1	25.5 ± 0.1
skvmfs	16.1 ± 0.1	24.4 ± 0.2	29.0 ± 9.2	39.1 ± 0.1
softvmfs	34.5 ± 2.2	76.8 ± 1.8	121.7 ± 0.1	178.8 ± 0.2
damns	78.6 ± 4.3	172.1 ± 7.4	252.5 ± 8.3	362.5 ± 17.9
davmfs	288.4 ± 10.0	671.4 ± 21.4	1050.7 ± 26.2	1584.0 ± 39.7
CLUTO[a]	18.6 ± 1.8	22.6 ± 1.7	25.1 ± 1.7	27.0 ± 1.7
co-cluster	20.9 ± 0.5	39.9 ± 1.0	62.8 ± 0.7	102.9 ± 0.8

[a]CLUTO is written in C whereas all the other algorithms are in Matlab.

4.6 Case Study Summary

At present, I am unaware of any other comprehensive comparative study of generative models for document clustering, or a comparison of such models with discriminative ones. I have successfully applied the unified framework in Chapter 3, instantiated with three different models, to document clustering. Empirical results across a large number of high dimensional text datasets highlighted the following trends: (a) the Bernoulli model is not appropriate for text clustering; (b) the von Mises-Fisher model often outperforms the multinomial model for clustering documents; (c) the algorithms using soft assignment run slower and only slightly improve the clustering performance for the Bernoulli and multinomial models; (d) the DA clustering improves the performance of vMF-based clustering significantly, especially on medium to small datasets.

Note that the *softvmfs* used in this work is not a full-fledged EM algorithm. Concurrent work at UT on an EM algorithm that allows different dispersion (κ) values for different clusters and lets EM re-estimate these values after each iteration, indicates that substantial gains can be achieved (Banerjee et al., 2003). Preliminary investigation indicates that the superior results are due to an annealing effect produced by using small initial κ's which are then automatically determined/annealed by the EM procedure. This is analogous to using very large initial variances for a mixture-of-Gaussians model. The results on using deterministic annealing for the mixture-of-multinomials and soft vMF-based clustering in this chapter also show that significant improvements can be obtained. Furthermore, DA is a more general technique and can be applied to discrete probabilistic models (e.g., multinomial) which have no tuning parameter similar to the κ in vMF models.

Chapter 5

Model-based Balanced Clustering

This chapter presents a general framework for adapting any generative (model-based) clustering algorithm to provide balanced solutions, i.e., clusters of comparable sizes. This represents a benefit of the unified model-based clustering framework in Chapter 3, which can be easily adapted to develop a general framework for new applications. Partitional, model-based clustering algorithms can be viewed as an iterative two-step optimization process—data assignment step and model estimation step. Instead of a maximum-likelihood assignment, a balance-constrained approach is used for the data assignment step. An efficient iterative bipartitioning heuristic is developed to reduce the computational complexity of this step and make the balanced data assignment algorithm scalable to large datasets. The superiority of this approach to regular maximum likelihood clustering are demonstrated on arbitrary-shaped 2-D synthetic data, high-dimensional text documents, and EEG time series.

The organization of this chapter is as follows. Section 5.1 discusses motivation for balanced model-based clustering. Section 5.2 presents a scalable, balanced clustering algorithm. Section 5.3 experimentally investigates the performance of the proposed balanced approach on several diverse types of datasets. Finally, section 5.4 concludes this chapter.

5.1 Motivation

Several real life data mining applications demand comparably sized segments of the data, irrespective of whether the natural clusters in the data are of comparable sizes or not (Ghosh, 2003). For example, a direct marketing campaign often starts with segmenting customers into groups of roughly equal size or equal estimated revenue generation, (based on market basket analysis, or purchasing behavior at a web site), so that the same number of sales teams, marketing dollars etc., can be allocated to each segment. In large retail chains, one often desires product categories/groupings of comparable importance, since subsequent decisions such as shelf/floor space allocation and product placement are influenced by the objective of allocating resources proportional to revenue or gross margins associated with the product groups (Strehl and Ghosh, 2003). Similarly, in clustering of a large corpus of documents to generate topic hierarchies, balancing greatly facilitates navigation by avoiding the generation of hierarchies that are highly skewed, with uneven depth in different parts of the "tree" hierarchy or having widely varying number of documents at the leaf nodes.

In addition to application requirements, balanced clustering is sometimes also helpful because it tends to decrease sensitivity to initialization and to avoid outlier clusters (highly under-utilized representatives) from forming, and thus has a beneficial regularizing effect. In fact, balance is also an important constraint for spectral graph partitioning algorithms (Dhillon, 2001; Kannan et al., 2000; Ng et al., 2002), which could give completely useless results if the objective function is just the minimum cut instead of a modified minimum cut that favors balanced clusters.

Unfortunately, k-means type algorithms, including the soft EM variant (Blimes, 1998), and BIRCH (Zhang et al., 1996), are increasingly prone to yielding imbalanced solutions as the input dimensionality increases. This problem is

exacerbated when a large (tens or more) number of clusters are needed, and it is well known that both hard and soft k-means invariably result in some near-empty clusters in such scenarios (Banerjee and Ghosh, 2002b; Bradley et al., 2000).

While not an explicit goal in most clustering formulations, certain approaches such as top-down bisecting k-means (Steinbach et al., 2000) tend to give more balanced solutions than others such as single-link agglomerative clustering. The most notable type of clustering algorithms in terms of balanced solutions is graph partitioning (Karypis and Kumar, 1998), but it needs $O(n^2)$ computation just to compute the similarity matrix. Certain online approaches such as frequency sensitive competitive learning (Ahalt et al., 1990) can also be employed for improving balancing. A generative model based on a mixture of von Mises-Fisher distributions has been developed to characterize such approaches for normalized data (Banerjee and Ghosh, 2002a).

The problem of clustering large scale data under constraints such as balancing has recently received attention in the data mining literature (Banerjee and Ghosh, 2002b; Bradley et al., 2000; Strehl and Ghosh, 2003; Tung et al., 2001). Since balancing is a global property, it is difficult to obtain near-linear time techniques to achieve this goal while retaining high cluster quality. Banerjee and Ghosh (2002b) proposed a three-step framework for balanced clustering: sampling, balanced clustering of the sampled data, and populating the clusters with the remaining data points in a balanced fashion. This algorithm has relatively low complexity of $O(N \log(N))$ but relies on the assumption that the data itself is very balanced (for the sampling step to work). Bradley, Bennett, and Demiriz (2000) developed a constrained version of the k-means algorithm. They constrained each cluster to be assigned at least a minimum number of data objects at each iteration. The cluster assignment subproblem was formulated as a minimum cost flow problem, which has a high ($O(N^3)$) complexity.

In this chapter, a balance-constrained approach is presented and built

upon the unified framework in Chapter 3. Based on a two-step view of the hard mk-means algorithms, a completely balanced data assignment subproblem is formulated and solved using a greedy heuristic at each iteration of the process. The heuristic results in an approximate solution to the subproblem in $O(K^2N + KN \log N)$ time. It can be easily generalized to handle partially balanced assignments and specific percentage assignment problems. Finally, a post-processing step can be attached to improve clustering quality in situations where strict balancing is not required.

There are several motivations behind this approach. First, probabilistic model-based clustering provides a principled and general approach to clustering (Banfield and Raftery, 1993). For example, the number of clusters may be estimated using Bayesian model selection, though this is not done in this dissertation. Second, the two-step view of partitional clustering is natural and has been discussed by Kalton et al. (2001). Finally, the post-processing step is motivated by the observation in (Bradley et al., 2000) that if the balancing constraint is too strict, then often distant data points are forceably grouped together leading to substantial degradation in cluster quality. It is observed that often a slight decrease in the amount of balance can buy substantial improvements in cluster quality, a favorable trade-off for many applications.

5.2 Balanced Model-based Clustering

5.2.1 Revisiting Model-based K-means

Recall that the objective function of the mk-means algorithm is

$$\log P(X|\Lambda) = \sum_n \log p(x_n|\lambda_{y_n}) \ . \tag{5.1}$$

Let z_{nk} be a binary assignment variable with value 1 indicating assignment of data object x_n to model k. The mk-means clustering can be re-formulated as the

following optimization problem

$$\begin{array}{ll} \max\limits_{\{\lambda,z\}} & \sum\limits_{n,k} z_{nk} \log p(x_n|\lambda_k) \\ \text{s.t.} & \sum_k z_{nk} = 1, \forall n;\ z_{nk} \in \{0,1\}, \forall n,k\ . \end{array} \tag{5.2}$$

Note that the step 2a in Fig. 3.4 corresponds to finding z_{nk}'s to maximize (5.2) with λ_k's fixed, and the step 2b corresponds to training λ_k's to solve (5.2) with z_{nk}'s fixed. This observation leads to a decomposition of (5.2) into two subproblems: a data assignment subproblem

$$\begin{array}{ll} \max\limits_{\{z\}} & \sum\limits_{n,k} z_{nk} \log p(x_n|\lambda_k) \\ \text{s.t.} & \sum_k z_{nk} = 1, \forall n;\ z_{nk} \in \{0,1\}, \forall n,k\ , \end{array} \tag{5.3}$$

and a model estimation subproblem

$$\max_{\{\lambda\}} \sum_{n,k} z_{nk} \log p(x_n|\lambda_k)\ . \tag{5.4}$$

As shown in the next section, this decomposition greatly facilitates the development and analysis of a balanced mk-means algorithm. Also note the formulation is generic in that one can plug in any probabilistic models into the subproblem (5.4).

5.2.2 Completely Balanced Mk-means Clustering

The completely balanced mk-means clustering problem can be defined as:

$$\begin{array}{ll} \max\limits_{\{\lambda,z\}} & \sum\limits_{n,k} z_{nk} \log p(x_n|\lambda_k) \\ \text{s.t.} & \sum_k z_{nk} = 1, \forall n;\ \sum_n z_{nk} = N/K, \forall k; \\ & z_{nk} \in \{0,1\}, \forall n,k. \end{array} \tag{5.5}$$

If N/K is not an integer, one can round it to the closest integer and make slight changes so that $\sum_{n,k} z_{nk} = N$ holds. This problem is again decomposed into two

Algorithm: completely balanced mk-means

Input: A set of N data objects $X = \{x_1, ..., x_N\}$ and model structure $\Lambda = \{\lambda_1, ..., \lambda_K\}$.

Output: Trained models Λ and a partition of data given by cluster identity vector $Y = \{y_1, ...y_N\},\ y_n \in \{1, ..., K\}$.

Steps:

1. Initialization: initialize model parameters Λ, cluster identity vectors Y and $z_{nk} = I(y_n = k)$;

2a. Model re-estimation: let λ_k's be the solution to problem (5.4);

2b. Sample re-assignment: let z's be the solution to problem (5.6) and set $y_n = \max_k z_{nk}$;

3. Stop if Y do not change, otherwise go back to Step 2a.

Figure 5.1: Completely balanced mk-means algorithm.

subproblems: the model estimation subproblem is the same as in (5.4), whereas the balanced data assignment subproblem becomes

$$\begin{aligned} \max_{\{z\}} \quad & \sum_{n,k} z_{nk} \log p(x_n|\lambda_k) \\ \text{s.t.} \quad & \textstyle\sum_k z_{nk} = 1, \forall n;\ \ \sum_n z_{nk} = N/K, \forall k; \\ & z_{nk} \in \{0, 1\}, \forall n, k. \end{aligned} \tag{5.6}$$

Fig. 5.1 describes a generic completely balanced mk-means algorithm that solves the problem (5.5). Its convergence is given by the following theorem.

Theorem 2 *If $p(x|\lambda)$ is bounded from above, the algorithm given in Fig. 5.1 will converge to a local maximum of the objective function in (5.5).*

Proof of this theorem is straightforward and similar to the proof of Theorem 1 in Section 3.2.

Note that the generic algorithm does not specify how to solve the balanced data assignment subproblem (5.6). It is an integer programming problem, which

is NP-hard in general. Fortunately, this integer programming problem is special in that it has the same optimum as its corresponding real relaxation (Bradley et al., 2000), which is a linear programming problem. The relaxed optimization problem is

$$\begin{aligned} \max_{\{z\}} \quad & \sum_{n,k} z_{nk} \log p(x_n|\lambda_k) \\ \text{s.t.} \quad & \textstyle\sum_k z_{nk} = 1, \forall n; \;\; \sum_n z_{nk} = N/K, \forall k; \\ & z_{nk} \geq 0, \forall n, k. \end{aligned} \tag{5.7}$$

The best known exact algorithm to solve this linear programming problem is an improved interior point method that has a complexity of $O(N^3K^3/\log(NK))$, according to (Anstreicher, 1999). The well-known simplex algorithm has exponential worst case time complexity but often exhibits polynomial expected time complexity (Fang and Puthenpura, 1993), which is around $O(N^3K)$ in this case.

To make this clustering algorithm scalable, I seek approximate solutions to (5.6) that can be obtained in time better than $O(N^2)$. There are a number of heuristics that can be used, such as the one-to-many stable matching algorithm, used by Banerjee and Ghosh (2002b) to populate balanced clusters (learned from a small sampled set of data points) with the remaining data objects. They named the algorithm "poly-stable" since the problem solved is in fact a polygamous version of the classic stable marriage problem. An alternative is an iterative greedy bipartitioning algorithm (Fig. 5.2) that assigns N/K data objects to one of the K clusters in a locally optimal fashion at each iteration. Both heuristics work well in preliminary experiments, but I choose the second one since it fits well within the optimization problem setting in this chapter. The completely balanced clustering algorithms using this greedy heuristic always converge in our experiments, though not monotonically.

The motivation behind this heuristic is that it solves (5.6) exactly for $K = 2$. In other words, if there are just two clusters, one simply sorts the difference vector $dv_n = \log p(x_n|\lambda_1) - \log p(x_n|\lambda_2), n = 1, ..., N$ in descending order and

Algorithm: Iterative greedy bipartitioning

Input: Log-likelihood matrix $w_{nk} = \log p(x_n|\lambda_k), n = 1, ..., N, \ k = 1, ..., K$.

Output: A partition matrix Z that satisfies $z_{nk} \in \{0, 1\}$, $\sum_k z_{nk} = 1, \forall n$ and $\sum_n z_{nk} = \frac{N}{K}, \forall k$.

Steps:

1. Initialization: set $j = 1$, $J = \{1, ..., N\}$ and $z_{nk} = 0, \forall n, k$;
2. Calculating log-likelihood difference vector: let $dv_n = w_{nj} - \max_{k>j} w_{nk}, \forall n \in J$ and $dv = \{dv_n\}_{n \in J}$;
3. Bipartitioning using sorted difference vector: sort dv in descending order and assign the top $\frac{N}{K}$ objects to cluster j, i.e., set $z_{nj} = 1, \forall n \in I$, where I is the set of indices corresponding to the top $\frac{N}{K}$ objects;
4. Stop if $j = K$, otherwise let $j = j + 1$, $J = J - I$ and go back to Step 2.

Figure 5.2: Iterative greedy bipartitioning algorithm.

assigns the first $N/2$ objects to cluster 1 and the second half to cluster 2. It is easy to show this gives a $\{\frac{N}{2}, \frac{N}{2}\}$ bipartition that maximizes (5.5).

For $K > 2$, a greedy bipartition is conducted at each iteration that separates the data objects for one cluster from all the others in such a way that the objective (5.1) is locally maximized. It is trivial to show that the j-th iteration of the algorithm in Fig. 5.2 gives a locally optimal $\{\frac{N}{K}, \frac{(K-j)N}{K}\}$ bipartition that assigns $\frac{N}{K}$ objects to the j-th cluster.

Let us now look at the time complexity of this algorithm. Let $N_j = \frac{(K+1-j)N}{K}$ be the length of the difference vector computed at the j-th iteration. Calculating the difference vectors takes $\sum_j N_j(K-j) \simeq O(K^2N)$ time and sorting them takes $\sum_j N_j \log N_j \simeq O(KN \log N)$ time. The total time complexity for this algorithm is $O(K^2N + KN \log N)$. The greedy nature of the algorithm stems from the imposition of an arbitrary ordering of the clusters using j. So one should investigate the effect of different orderings. In the experiments, the ordering is done at random in each experiment, multiple experiments are run and the variation in results is inspected. The results exhibit no abnormally large variations and suggest that the effect of ordering is small.

The algorithm can be easily generalized to solve the data assignment problem with specific balance constraints. For example, if prior knowledge about the partition percentages (e.g. a $\{20\%, 20\%, 30\%, 30\%\}$ partition) is available, one can easily build the numbers into the algorithm and assign a corresponding number of data objects to each cluster. Or if one just wants each cluster to have at least $m(< \frac{N}{K})$ objects, i.e. similar to (Bradley et al., 2000), one can assign just the top m objects at each iteration and use ML assignment for the remaining data. There are certainly many other variations (that may be useful in practice), but they are not the focus of this dissertation.

5.2.3 Refinement Step

This step is really an optional step, depending on what kind of results are desired. If approximate rather than exact balanced solutions are acceptable, then a minor refinement can be used to improve cluster quality. This is achieved by letting the results from completely balanced mk-means serve as an initialization for the regular mk-means. Since the regular mk-means has relative low complexity of $O(KN)$, this extra overhead is low. The experiments reported in this chapter reflect a "full" refinement in the sense that the regular mk-means in the refinement step is run until convergence. Alternatively, partial refinement such as one round of ML re-assignment can be used and is expected to give an intermediate result between the completely balanced one and the "fully" refined one. In the experimental results, intermediate results are not shown but they will be bounded from both sides by the completely balanced and the "fully" refined results.

The refinement step can be viewed from a second perspective— results from completely balanced clustering serve as an initialization to regular mk-means clustering. From this point of view, the completely balanced data assignment results in much better initial clusters than random initialization.

5.3 Clustering Results and Discussions

5.3.1 Clustering Models

In this section, I briefly review the four generative models used in the experiments. All these models have been introduced in Section 2.2. The first one is the spherical Gaussian, which is the model behind the standard k-means. The two models used to model high-dimensional sparse document vectors, are von Mises-Fisher and multinomial, respectively. The last one, used to cluster sequences, is the standard HMM. I name the latter three generalized k-means algorithms k-vMFs,

k-multinomials and k-HMMs, respectively.

K-means

For k-means clustering, the average log-likelihood (ALL) objective is

$$ALL_{\text{k-means}} = -\frac{1}{N}\sum_{n}\|x_n - \mu_{y_n}\|^2. \tag{5.8}$$

Note here the contribution from the variance parameter is omitted from the log-likelihood since it is assumed to be a constant across different clusters. It can be seen that this is exactly a negative version of the sum-squared error objective that the k-means algorithm minimizes.

K-vMFs

The objective function evaluated for this model is the average log-likelihood with the contribution from the dispersion parameter κ omitted, which then becomes exactly the average cosine similarity of all data objects and their cluster means.

$$ALL_{\text{k-vMFs}} = \frac{1}{N}\sum_{n} x_n^T \mu_{y_n} \ . \tag{5.9}$$

K-multinomials

The average log-likelihood evaluated for this model is

$$ALL_{\text{k-multinomials}} = \frac{1}{N}\sum_{n}\sum_{l} x_n^{(l)} \log P_n^{(l)} \ , \tag{5.10}$$

where $x_n^{(l)}$ is the l-th dimension of x_n.

K-HMMs

The ALL objective used for the k-HMMs algorithm is

$$ALL_{\text{k-HMMs}} = \frac{1}{N}\sum_{n}\frac{1}{T_n}\log p(x_n|\lambda_{y_n}) \ , \tag{5.11}$$

where T_n is the length of sequence x_n.

For each of the above models, I compare three versions of the clustering algorithm in the experiments: a regular version, a balanced version and a balanced+refinement version. For simplicity, the balanced algorithms for four different models are named bk-means, bk-vMFs, bk-multinomials and bk-HMMs, respectively, and the refined ones are called refined bk-means, refined bk-vMFs, ..., etc. As a reminder, the balanced version generates completely balanced clustering and the refined version attaches a post-processing phase to the balanced version. Finally, I emphasize that this balanced approach is built on the general model-based clustering framework and applies to any application for which appropriate models exist, for the balance constraint is completely independent of the model re-estimation part.

5.3.2 Results on Synthetic Data

I first tested the balanced k-means algorithm on a synthetic but difficult dataset—the t4 dataset (Fig. 5.3) included in the CLUTO toolkit (Karypis, 2002). There are no ground truth labels for this dataset but there are six natural clusters plus a lot of noise according to human judgment. The graph partitioning algorithm (Karypis et al., 1999; Karypis, 2002) that can identify all the six natural clusters uses a two-step partitional+hierarchical approach. It partitions the data into a large number of clusters and then merges them back to a proper granularity level. Graph partitioning algorithms can often lead to balanced and high quality clusters, thus fit in well in the first step. The standard k-means algorithm has better complexity but cannot fulfill the requirement for the first step of the hybrid clustering. The refined bk-means algorithm provides an intermediate solution that can generate balanced, high quality clustering while still possessing relatively low complexity.

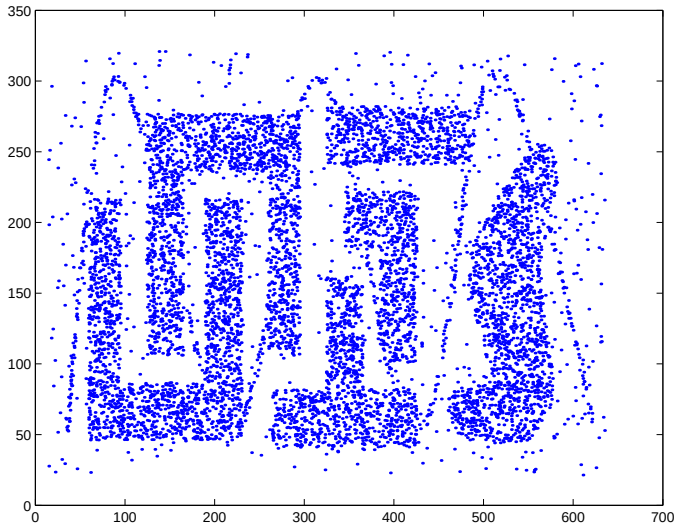


Figure 5.3: The t4 dataset.

Fig. 5.4(a) shows the balance results[1] and Fig. 5.4(b) the average log-likelihood results, for different number of clusters. All the results are averaged over ten experiments with the bars at each data point indicating ± 1 standard deviation in ten experiments. The balance performance of standard k-means deteriorates significantly as the number of clusters increases. The bk-means algorithm always delivers perfectly balanced clusterings but pays in terms of ALL objective. In contrast, the refined bk-means approach achieves very balanced clusterings as well as dramatically better objectives than the regular k-means. In this case, the refinement step seems to be able to keep the balance of the clusterings from bk-means and improve the ALL objective simultaneously. Further more, the ALL performance gap between the refined bk-means and the standard k-means widens as the number of clusters increases.

Fig. 5.5(a)&(b) show a typical clustering result for $K = 30$. It can be seen that the standard k-means produces many clusters that mix data points from

[1]I reused the symbols and colors in the figure, but each cluster is represented by a unique combination of symbol and color.

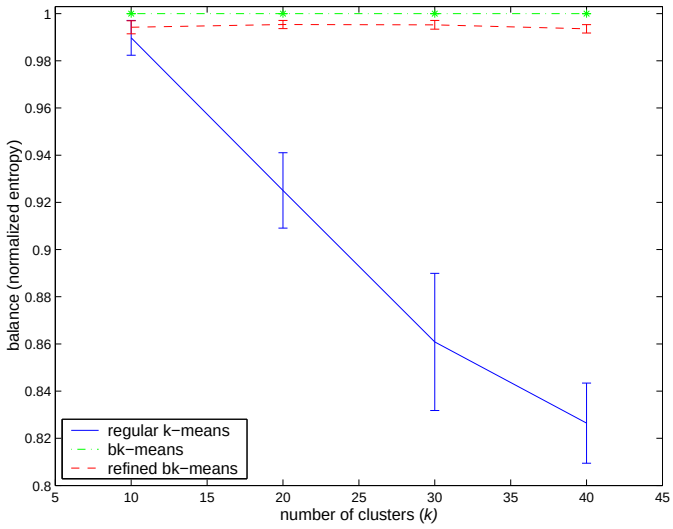
different natural clusters, whereas the refined bk-means gives much cleaner and more balanced results. Fig. 5.6(a)&(b) show the histogram distribution of cluster sizes for the results in Fig. 5.5(a)&(b). The standard k-means produces many empty clusters and large variations in cluster sizes, whereas the refined bk-means gives much more balanced clusters. In addition, the refined bk-means achieves a much better local solution (with an ALL of -620.9 vs. -1237.1 for the standard k-means).

5.3.3 Results on Real Text Data

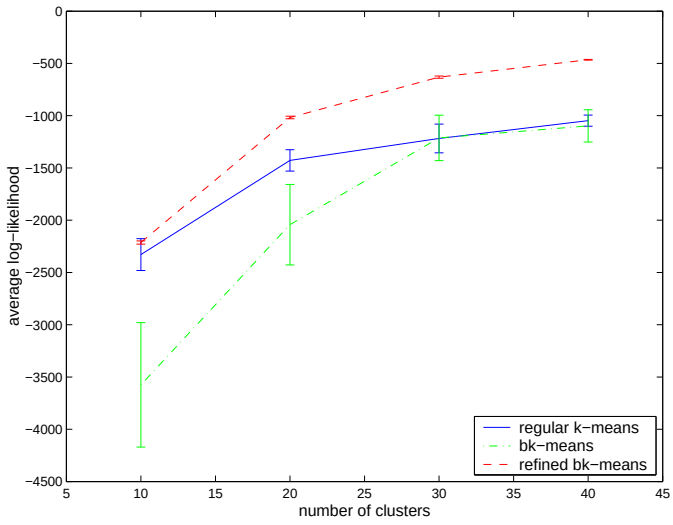
I used the 20-newsgroups (*NG20*), mini 20-newsgroups (*mini20*), and *classic* datasets for empirical performance analysis on text data. The *NG20* and *classic* datasets have been introduced in Section 4.3. The *mini20* dataset is a random sample from *NG20*, with 100 messages from each category. After the same preprocessing step, the resulting dataset consists of 1,998 documents in 10,633 dimensional vector space. This dataset has been used by Nigam (Nigam, 2001) for text classification and is included in this dissertation to evaluate the performance of different models on small document collections. Both the *NG20* and *mini20* datasets contain 20 completely balanced clusters. The *classic* dataset has a slightly unbalanced set of categories, with a normalized entropy of 0.92.

Two types of models, vMFs and multinomials, are used. For vMF models, again log(IDF)-weighted (and normalized) document vectors are used. For each model type, I compare the balanced model-based clustering with regular ML clustering in terms of balance, objective value and mutual information with original labels, over different number of clusters.

Fig. 5.7–5.15 show the results on the *NG20*, *mini20*, and *classic* datasets. with results for multinomial models in the top row and those for vMF models in the bottom row. Fig. 5.7, 5.10, and 5.13 show the balance results (normal-

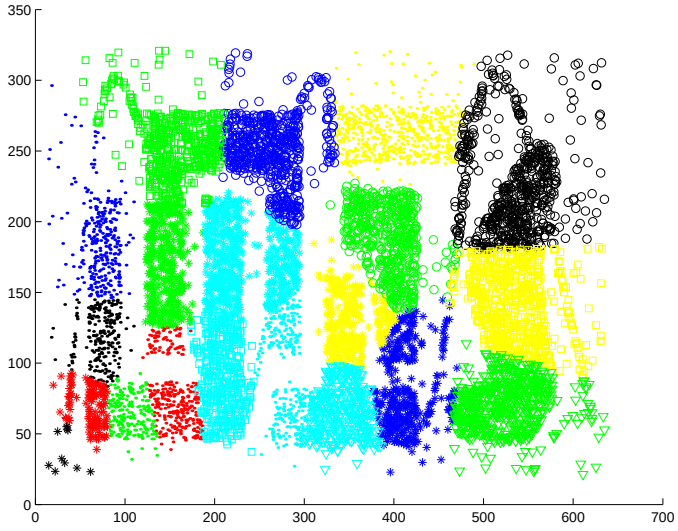


(a)

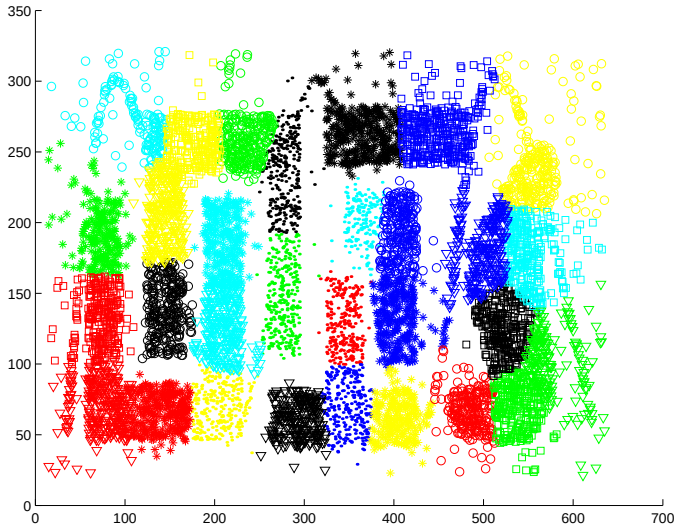


(b)

Figure 5.4: Results on the *t4* dataset: comparison of (a) normalized entropy results and (b) average log-likelihood results.

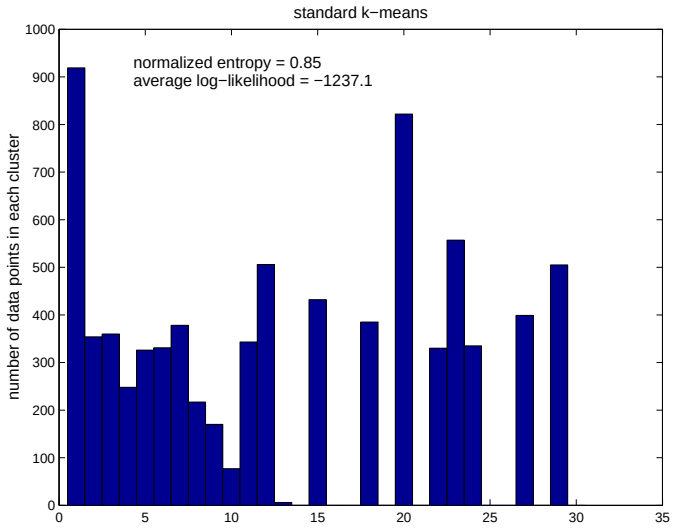


(a)

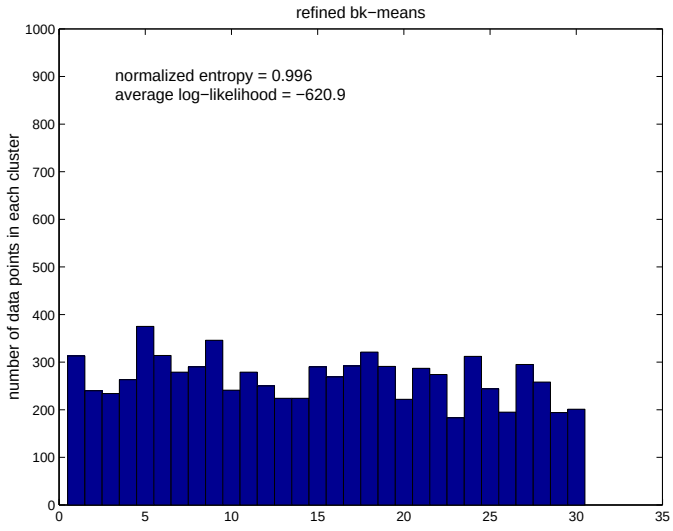


(b)

Figure 5.5: Results on the *t4* dataset: a typical 30-cluster partition from (a) the standard k-means algorithm and (b) the refined bk-means algorithm.



(a)



(b)

Figure 5.6: Results on the *t4* dataset: histogram distribution of cluster sizes for (a) the standard k-means algorithm and (b) the refined bk-means algorithm.

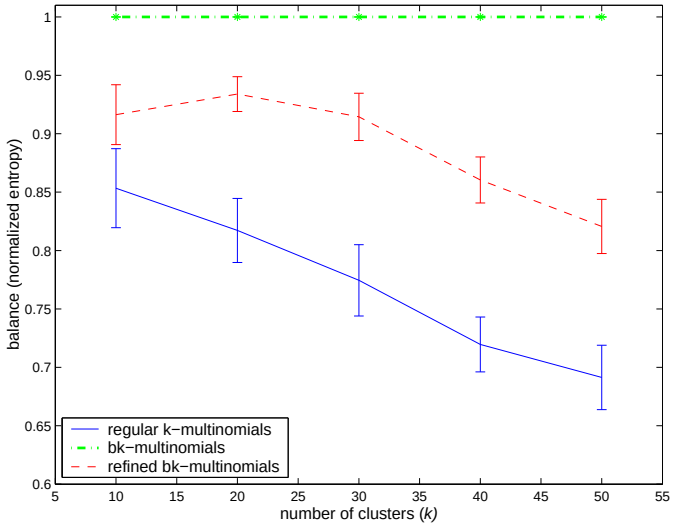
ized entropies). Fig. 5.8, 5.11, and 5.14 show the average log-likelihood results (normalized entropies). Fig. 5.9, 5.12, and 5.15 show the NMI results (normalized entropies). All results are shown as *average* $\pm$ 1 *standard deviation* over 20 experiments.

In all cases, the completely balanced clustering algorithms produce worse clusterings in terms of both the ALL and NMI measures since that perfect balancing is too strict a constraint. But the balanced clustering algorithms, with refinement, perform either comparably or significantly better than regular ML clustering in terms of both ALL and NMI results, and provide significantly more balanced clusterings than the regular methods.

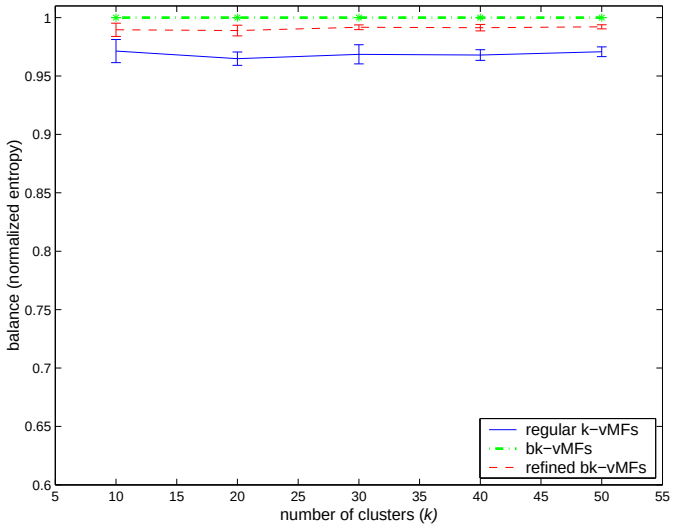
Comparing multinomial models with vMF ones, one can see that the vMF-based algorithms produce much more balanced clusterings for all datasets. In terms of NMI, k-multinomials and k-vMFs work comparably for large collections (*NG20* and *classic*, 1,000+ documents per cluster) and k-multinomials tends to be better for large number of clusters. For small collections (*mini20*, 100 documents per cluster), the vMF-based algorithms perform significantly better than multinomial ones. These results are in accord with those seen in Section 4.5.

5.3.4 Results on EEG Time-Series

The EEG data from the UCI KDD archive (http://kdd.ics.uci.edu) arose from a large study to examine EEG correlates of genetic predisposition to alcoholism. There are two groups of subjects in the study: alcoholic and control. Each subject is exposed to stimuli that are pictures of objects chosen from the 1980 Snodgrass and Vanderwart picture set. There are three different types of stimuli. The data contains measurements, sampled at 256 Hz for 1 second, from 64 electrodes placed on the human scalp. I extracted from the archive a subset called *EEG12* (Zhong and Ghosh, 2002a), that contains 20 measurements for two subjects—one alcoholic

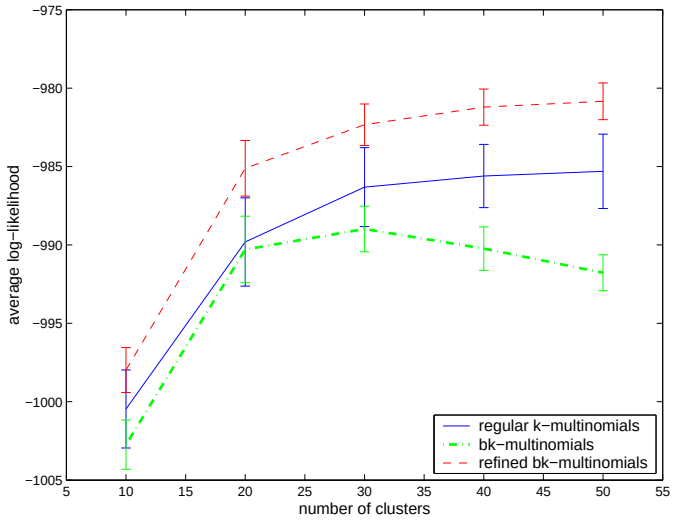


(a)

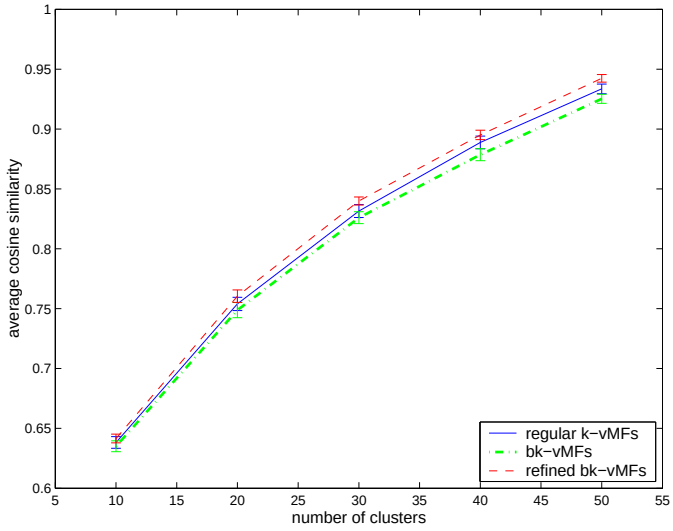


(b)

Figure 5.7: Results on the *NG20* dataset: balance results for (a) multinomial models and (b) vMF models.

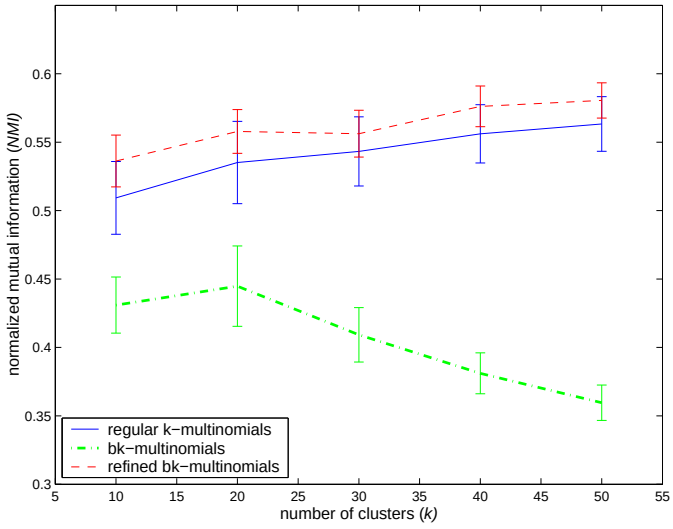


(c)

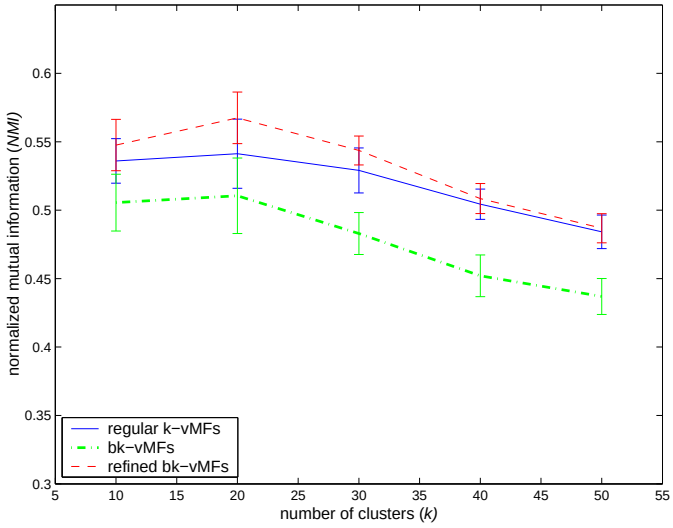


(d)

Figure 5.8: Results on the *NG20* dataset: (a) average log-likelihood results for multinomial models and (b) average cosine similarity results for vMF models.

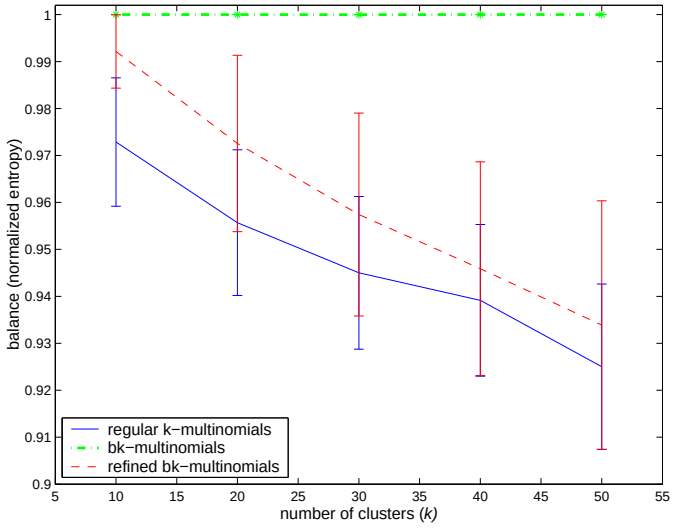


(e)

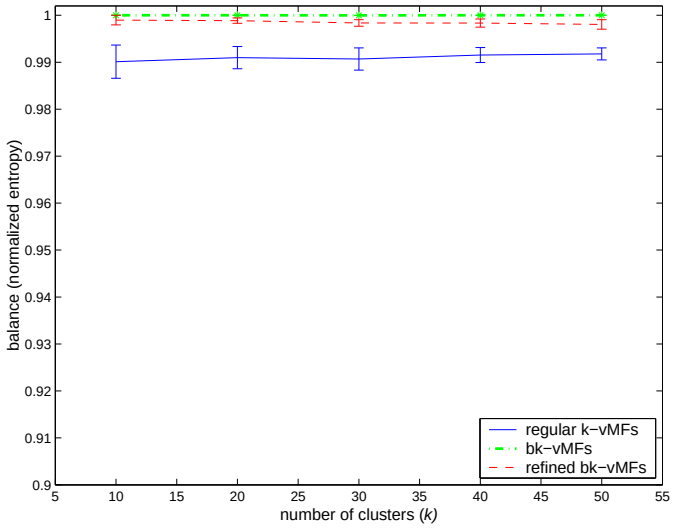


(f)

Figure 5.9: Results on the *NG20* dataset: normalized mutual information results for (a) multinomial models and (b) vMF models.

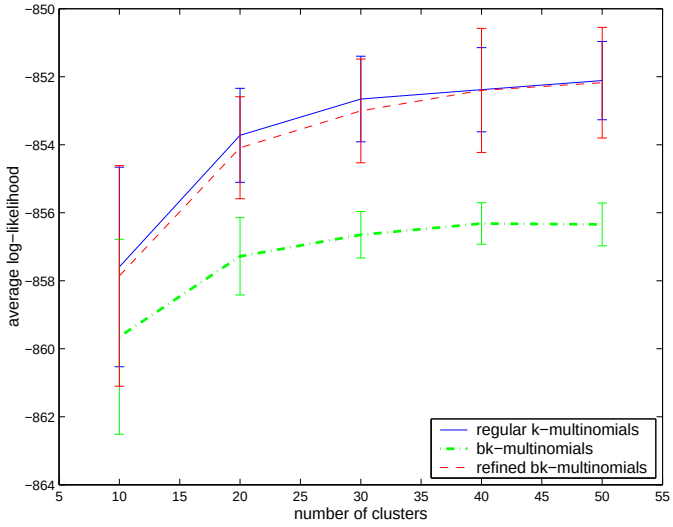


(a)

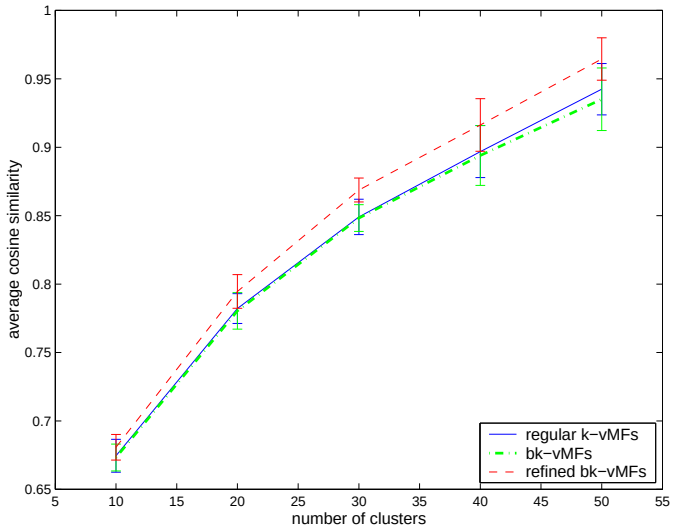


(b)

Figure 5.10: Results on the *mini20* dataset: balance results for (a) multinomial models and (b) vMF models.

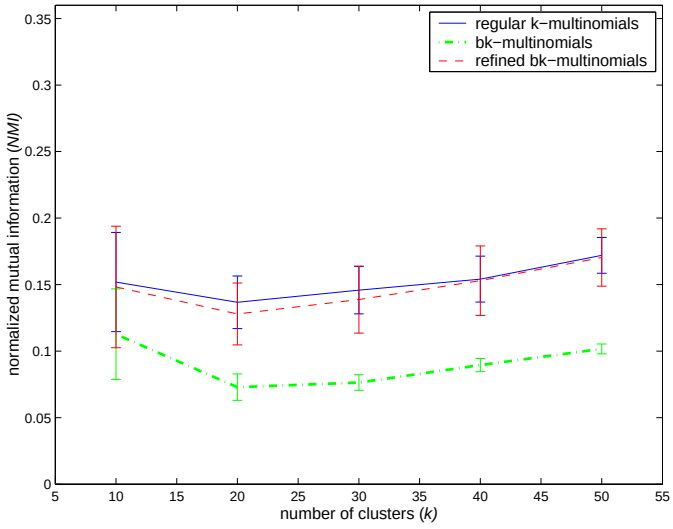


(a)

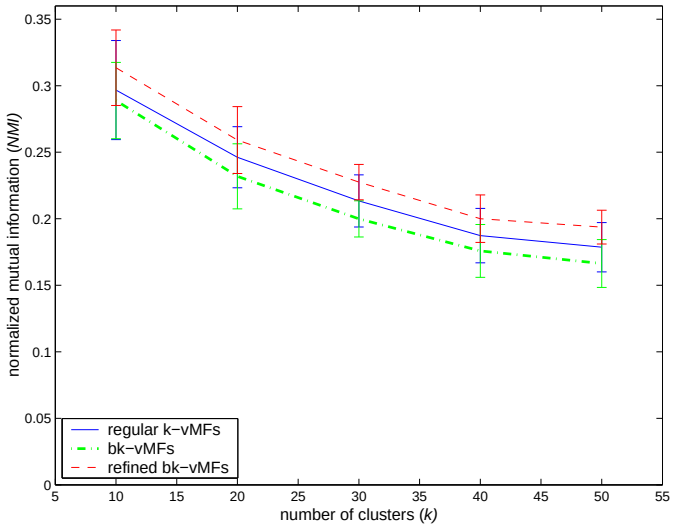


(b)

Figure 5.11: Results on the *mini20* dataset: (a) average log-likelihood results for multinomial models and (b) average cosine similarity results for vMF models.

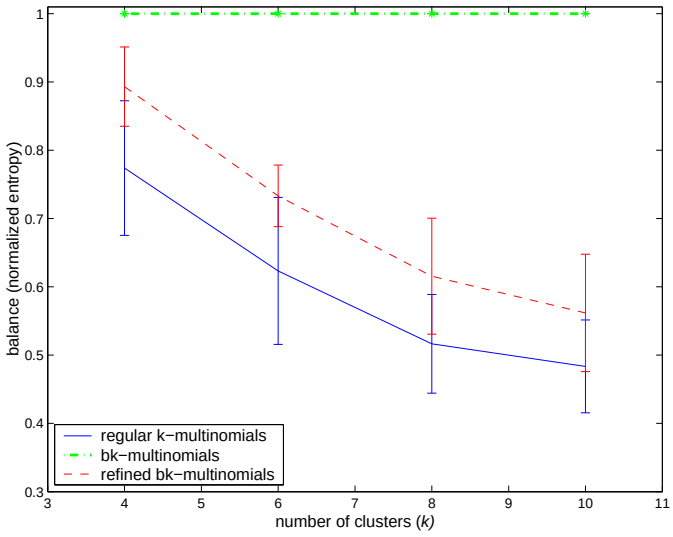


(a)

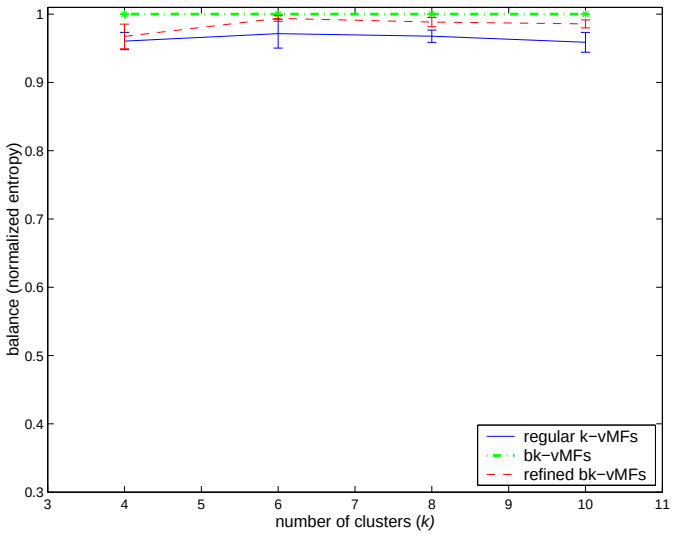


(b)

Figure 5.12: Results on the *mini20* dataset: normalized mutual information results for (e) multinomial models and (f) vMF models.

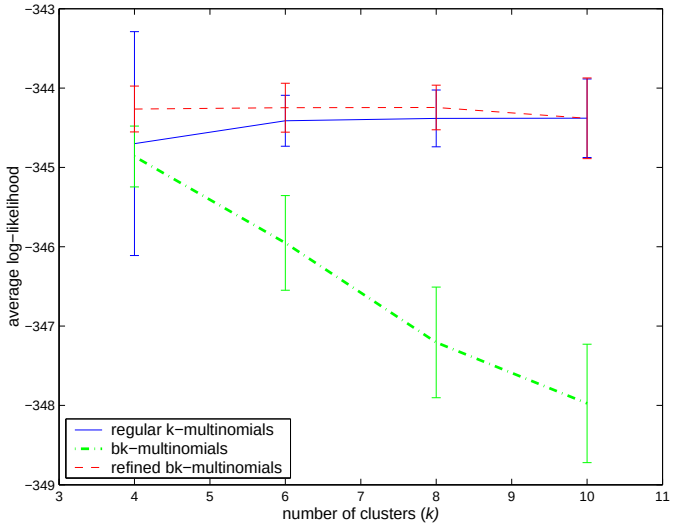


(a)

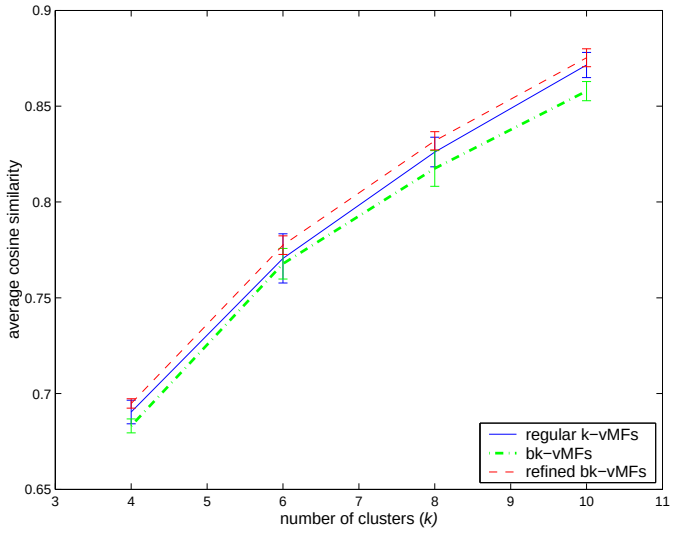


(b)

Figure 5.13: Results on the *classic* dataset: balance results for (a) multinomial models and (b) vMF models.

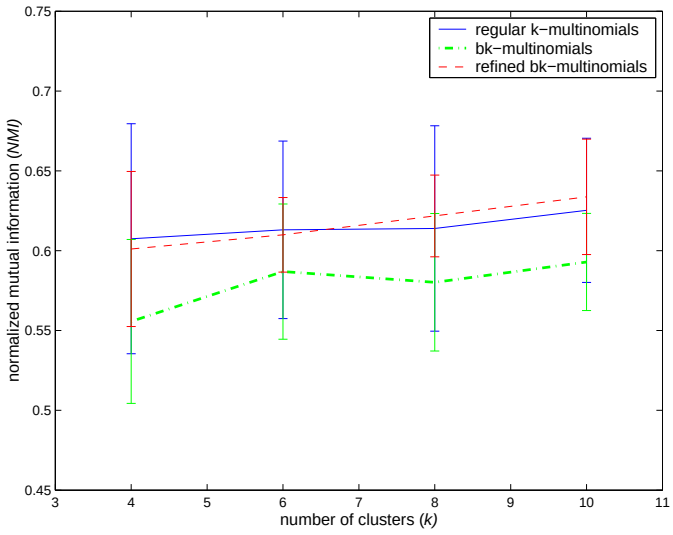


(a)

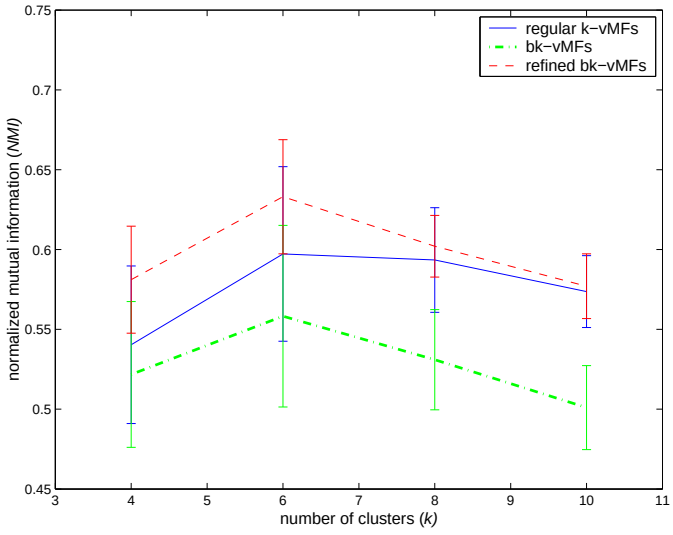


(b)

Figure 5.14: Results on the *classic* dataset: (a) average log-likelihood results for multinomial models and (b) average cosine similarity results for vMF models.



(a)



(b)

Figure 5.15: Results on the *classic* dataset: normalized mutual information results for (a) multinomial models and (b) vMF models.

and one control, for each type of three stimuli types, from 2 electrodes (F4, P8). As a result, the *EEG12* dataset contains 120 data objects and each data object is a pair of sequences of length 256.

When modeling EEG time series, I set the dimension of the observation vector to be the number of EEG channels, and use an HMM with bivariate Gaussian observations and five hidden states. I evaluate the performance of the regular k-HMMs, bk-HMMs and refined bk-HMMs on the *EEG12* dataset with 6 clusters. The results are shown in Table 5.1. Similar to the results from previous experiments, bk-HMMs generates lower quality clusters than the regular k-HMMs. But the refined bk-HMMs significantly outperforms the regular k-HMMs in terms of both balance and ALL measures, based on t-tests at $p = 0.05$ level. In addition, it leads to slightly better NMI values.

Table 5.1: Clustering results on *EEG12* with 6 clusters

	N_{entro}	ALL	NMI
regular k-HMMs	0.90 ± 0.05	-113.4 ± 1.0	0.32 ± 0.03
bk-HMMs	1.0 ± 0	-114.2 ± 1.6	0.31 ± 0.04
refined bk-HMMs	0.97 ± 0.02	-112.4 ± 0.6	0.33 ± 0.03

5.4 Summary of Balanced Clustering

I have presented a general framework for scalable, balanced model-based clustering that comes from an analysis of the two-step optimization process embedded in any model-based partitional clustering algorithm. The balanced approach is applicable to any domain for which good probabilistic models exist. I have used an efficient greedy heuristic to solve the balanced data assignment subproblem in $O(KN(K + \log N))$ time, and employed a refinement phase to improve the quality of clusters generated by the completely balanced mk-means algorithm. Finally,

I have demonstrated the superiority of this two-phase clustering algorithms to regular ML clustering using both synthetic and real (text and time-series) data.

In the hybrid partitional-hierarchical clustering in Chapter 6, an initial balanced flat partition is often desired as the starting point for subsequent hierarchical clustering. The experimental results on the synthetic dataset suggest that the balanced clustering algorithm might serve well for this purpose. As we will see in Chapter 6, it indeed helps the hybrid clustering approaches in some datasets.

I have focused on hard clustering algorithms in this chapter. It would be interesting to investigate the extensions to soft clustering. Also, there are several methods for estimating the number of clusters in a model-based clustering framework (Fraley and Raftery, 1998). The interaction of balanced clustering and model selection strategies can be investigated.

Spectral clustering has been a hot area of clustering research and often reduces to using standard k-means to search for appropriate clusters in the eigenspace of either the original vector data (Kannan et al., 2000) or the similarity graph constructed from the original data (Ng et al., 2002). The balanced k-means may be used to replace the standard k-means algorithm and get more balanced and hopefully improved spectral clustering results.

Chapter 6

Model-based Hybrid Clustering

This chapter presents a novel hybrid clustering methodology that combines the advantages of model-based partitional clustering and model-based hierarchical clustering algorithms.

The hybrid idea is simple, but the effect is significant. My contribution for this chapter is a theoretical justification of the hybrid model-based clustering algorithms based on the framework developed in Chapter 3 and a detailed empirical demonstration of their strong performance in several distinct applications. Section 6.1 gives the motivation behind the hybrid clustering approach. Section 6.2 presents a theoretical analysis of the proposed hybrid model-based clustering algorithms. Section 6.3 shows experimental results on arbitrary-shaped synthetic data, text documents, EEG time series, and gene expression time series. Conclusion and discussion are presented in Section 6.4.

6.1 Motivation

Mainly due to their high computational complexity, the hierarchical model-based clustering algorithms are not as popular as the partitional ones for large datasets despite their hierarchical visualization advantage. To reduce the complexity, Vaithyanathan and Dom (2000) adopted the simple idea of building a cluster

hierarchy from the results of a partitional clustering, which results in a two-step method for clustering documents. This hybrid idea, however, has not been widely recognized and formally analyzed. The purpose of this chapter is to advocate this simple idea by a detailed analysis. A useful variation of this hybrid idea is also presented.

Before delving into the details, let us take a look at other related work. A hybrid approach was used by Cutting et al. (1992) in the "Scatter/Gather" approach, for reducing computational complexity. They used the HAC algorithm on a small sampled dataset and fed the resulting clusters as an initialization to a subsequent k-means algorithm, which is run on the entire dataset. The same idea was explored by Meila and Heckerman (2001) to supply initial cluster centers for a second EM clustering step.

Another related work is the classic ISODATA algorithm (Hall and Ball, 1967), which refines the results of k-means by splitting and merging. Clusters are merged if either the cluster sizes are smaller than a certain threshold or the centers of two clusters are closer than a certain threshold. A cluster is split into two if its standard deviation exceeds a predefined value. This method needs several user-specified thresholds and assumes spherical clusters.

The multi-level graph partitioning algorithms (Karypis et al., 1999; Karypis, 2002) are similarity-based clustering approaches with a hybrid flavor. For example, the CLUTO toolkit (Karypis, 2002) allows the user to specify a large number of initial clusters, which are agglomeratively merged into the final desired number of clusters.

6.2 Hybrid Model-based Clustering

The hybrid clustering idea, illustrated in Fig. 6.1, is a "reverse" of the "Scatter/Gather" approach (Cutting et al., 1992). Data objects are first clustered into

K_0 (greater than the natural number of clusters K, which may be unknown) groups, that is, into finer granularity, using a partitional method such as the EM or mk-means algorithm. For model-based clustering, this step generates K_0 models and can be viewed as a compression/coarsening of the data into K_0 cluster representatives (Jain et al., 1999). In the second step, the HAC algorithm is used, starting from the K_0 clusters, to iteratively merge the two closest clusters until all data objects are in one cluster or the process is stopped by a user. This hybrid approach returns a series of nested clusterings that can be either interactively analyzed by a user or evaluated with some optimization criterion. This idea is not limited to model-based clustering but I focus on model-based hybrid clustering in this chapter. In the model-based clustering context, the hybrid clustering concept can be demonstrated using a bipartite graph view, as shown in Fig. 6.2. Fig. 6.2 differs from Fig. 3.3 in just the initial number of models/clusters used (K_0 vs. N). This difference has a big effect in the computational complexity as explained next.

Fig. 6.3 shows a generic model-based hybrid clustering algorithm based on the idea in Fig. 6.1. There are several benefits resulting from this idea. First, it can alleviate to some extent the initialization problem associated with partitional methods since when partitioned into finer granularity the quality of clusters (e.g., purity) tends to be higher. Consider the extreme case when $K_0 = N$, there is no initialization problem since every initial cluster is 100% pure (but the computational complexity is of course high). Basically, the hybrid approach aims to balance such a tradeoff by using a K_0 that is larger than K but still far smaller than N. By doing so, one can get around the high computational complexity of the HAC algorithm by running it on K_0 clusters instead of starting from N singleton clusters. This hybrid approach has a complexity of $O(K_0^2 N)$ or $O(K_0^3 N)$, depending on whether a hard or soft partitional method is used, but is still linear in N. The initial number K_0 can be set to be a constant times K. Finally, the

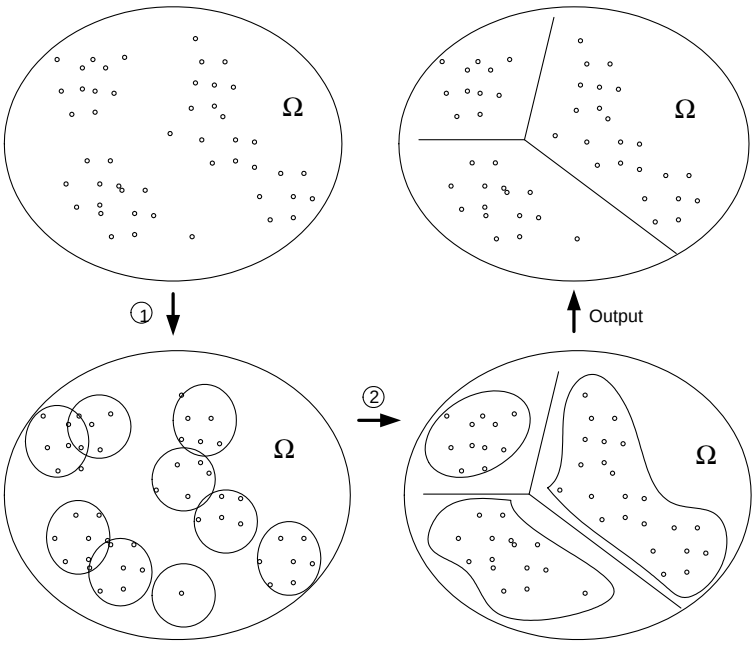


Figure 6.1: A hybrid clustering methodology. In step 1, a partitional clustering technique is used to cluster the whole data into many small groups (fine granularity); then in step 2, an HAC method is used to iteratively merge the large number of small clusters back into a desired small number of clusters (coarse granularity).

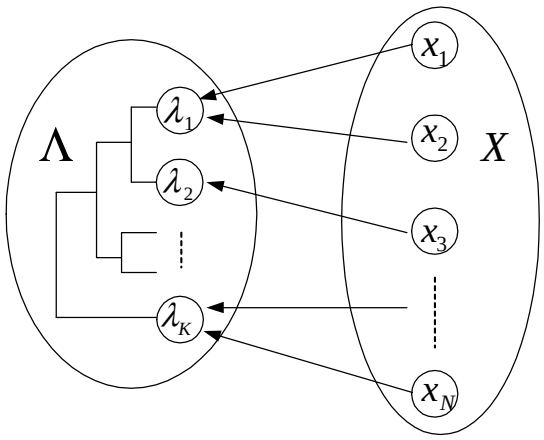


Figure 6.2: A bipartite graph view of model-based hybrid partitional-hierarchical clustering.

Algorithm: model-based hybrid partitional-hierarchical clustering

Input: A set of N data objects $X = \{x_1, ..., x_N\}$, model structure λ and the depth of hierarchy K_0.

Output: A K_0 level cluster (model) hierarchy and hierarchical partition of the data objects, with k clusters at the k-th level.

Steps:

1. Flat partitional clustering: partition data objects into K_0 clusters using one of the model-based partitional clustering algorithms discussed in Section 3.2;

2a. Distance calculation: compute pairwise inter-cluster distances using one of the measures defined in (3.13)-(3.16), and find the closest cluster pair;

2b. Cluster merging: merge the two closest clusters (assume they are cluster k and cluster j) and re-estimate a model for the merged data objects $X_k = X_k \cup X_j$, i.e., let $\lambda_k = \arg\max_{\lambda} \sum_{x \in X_k} \log p(x|\lambda)$;

3. Stop if all data objects have been merged into one cluster or user-specified number of clusters is reached, otherwise go back to Step 2a.

Figure 6.3: Model-based hybrid partitional-hierarchical clustering algorithm.

hybrid approaches tend to produce high quality clustering results, as shown in the empirical studies in Section 6.3.

Next I introduce a special version of this hybrid algorithm—a hierarchical meta-clustering algorithm, which improves efficiency and has some attractive properties.

Hierarchical Meta-Clustering

Let us first introduce a composite model $\lambda_{jj'} = \{\lambda_j, \lambda_{j'}\}$ for the cluster merged from cluster j and j' and define the likelihood of a data object x given this model

as

$$p(x|\lambda_{jj'}) = \max\{p(x|\lambda_j), p(x|\lambda_{j'})\} \ . \tag{6.1}$$

Let us call λ_j and $\lambda_{j'}$ children of the composite model $\lambda_{jj'}$. For the set of data objects in the merged cluster $X_{jj'} = X_j \cup X_{j'}$, we then have

$$\log P(X_{jj'}|\lambda_{jj'}) = \log P(X_j|\lambda_j) + \log P(X_{j'}|\lambda_{j'}) \ . \tag{6.2}$$

Furthermore, I define the distance between two composite models

$$\lambda_a = \{\lambda_{a_1}, \lambda_{a_2}, ...\}$$

and

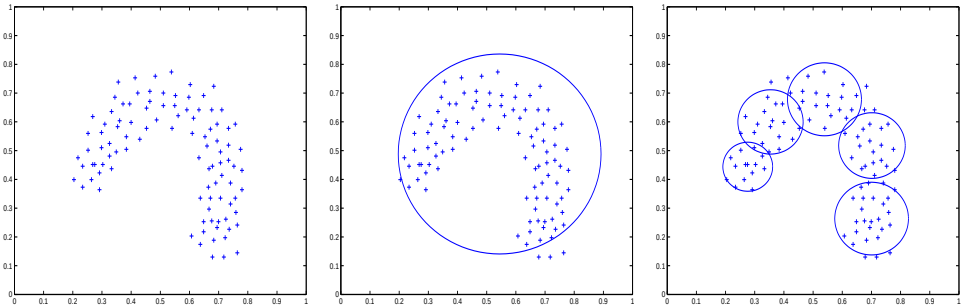
$$\lambda_b = \{\lambda_{b_1}, \lambda_{b_2}, ...\}$$

to be

$$D(\lambda_a, \lambda_b) = \min_{\substack{\lambda \in \lambda_a \\ \lambda' \in \lambda_b}} D(\lambda, \lambda') \ . \tag{6.3}$$

Two immediate benefits result from the above design. First, no model (parameter) re-estimation is needed after merging two clusters, since a composite model is simply represented by the parameters of its children. From (6.2), it is clear that the cluster merging does not change the likelihood $P(X|\Lambda)$, which also means that the Ward's distance (3.12) cannot be used in this case. Second, a composite model can be used to characterize complex clusters that a single model represents poorly. For example, a (rotated) u-shape cluster (Fig. 6.4) cannot be accurately modeled by a single Gaussian but can be approximated by a mixture of Gaussian distributions. Using a single Gaussian model loses the u-shape structure of the cluster, as shown in Fig. 6.4(b).

Using the composite models defined in (6.1) and the inter-cluster distances in (6.3), one can get a hierarchical meta-clustering algorithm, which is equivalent to treating each initial cluster as a meta-object and applying the traditional single-link hierarchical clustering algorithm to group the meta-objects. Obviously,



(a) (b) (c)

Figure 6.4: (a) A u-shape cluster. (b) Using a single spherical Gaussian model for the cluster. (c) Using a union of five spherical Gaussian models for the cluster. (Each circle shows the isocontour of a Gaussian model at two times the standard deviation.)

nothing prevents us from using a different hierarchical method (e.g., complete-link, average-link, etc.) to cluster the meta-objects, by suitably modifying the measure definition in (6.3). Each hierarchical method can be desirable in different applications. I call the hierarchy resulting from the hierarchical meta-clustering *meta-cluster hierarchy*.

The hierarchical meta-clustering algorithms can be seen as a combination of model-based partitional clustering and discriminative hierarchical methods. Compared to using a single complex model, I favor this strategy of merging simple models to form complex clusters since a single complex model is difficult to define and to train. For example, what is the distribution for the u-shape cluster in Fig. 6.4(a)? Furthermore, it is almost impossible to avoid poor local solutions even if such a complex distribution can be defined.

Balanced partitional step

To make the two-step hybrid approaches a success, it is often helpful to produce approximately balanced clusters in the first step. This may not be immediately clear, so let us again look at the u-shape cluster in Fig. 6.4(a) as an example.

Suppose the data is divided into five clusters in the flat clustering step. Using the k-means algorithm, one may get a very unbalanced clustering that contains one big cluster as in Fig. 6.4(b) and four other near empty clusters (not shown), or a balanced solution as in Fig. 6.4(c). While merging the five clusters back to one in either solution leads to the same set of data objects, the latter solution is preferred since it provides a useful hierarchy, disclosing the u-shape structure of the cluster. Therefore, the balanced model-based clustering algorithms developed in Chapter 5 are used to improve the performance of the hybrid approach (Section 6.3.1).

6.3 Experimental Results and Discussions

In this section, I demonstrate the superiority of the hybrid clustering approach as well as the generality of the model-based clustering framework via four case studies. For each case study, I shall compare the same set of generic algorithms studied in Section 3.2, instantiated with an appropriate model, with the corresponding hybrid approach. Specifically, the mixture of Gaussian and von Mises-Fisher distributions have been used for clustering 2-D arbitrary-shaped vector data and high-dimensional documents, respectively; Hidden Markov models and Markov chains have been employed for clustering EEG and gene expression time series, respectively.

6.3.1 Results on Synthetic 2-D Spatial Data

The hybrid algorithm is tested on two synthetic but difficult datasets: the *d4* dataset (Fig. 6.5), which contains 200 artificially generated data points, and the *t4* dataset (Fig. 5.3) included in the CLUTO toolkit (Karypis, 2002), which contains 8000 data points. There are no ground truth labels for these datasets but there are clearly four natural clusters for the *d4* dataset and six for the *t4* dataset (plus

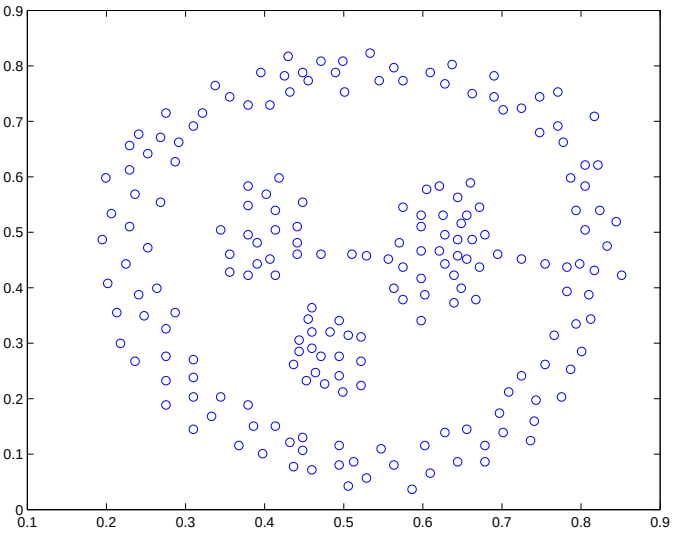


Figure 6.5: Synthetic d4 dataset.

some noise/outliers) according to human judgment. It is obvious that most of these natural clusters are not Gaussian. In fact, it is difficult to propose a model that fits all of the arbitrary-shaped 2-D clusters.

At first sight, one would probably turn to graph partitioning approaches to get good clustering results. Indeed, both the standard k-means and HAC algorithms fail miserably on these two datasets, whereas the spectral clustering algorithms (Kannan et al., 2000; Ng et al., 2002) identify the four natural clusters in *d4*, and a sophisticated graph partitioning approach (Karypis et al., 1999; Karypis, 2002) produces all six natural clusters in *t4*. The sophisticated graph partitioning algorithm first partitions the data into a large number of clusters and then merges (and refines) them back to a proper granularity level. In this section, I demonstrate that the same high quality clusters can be achieved more efficiently with the proposed hybrid model-based clustering algorithms.

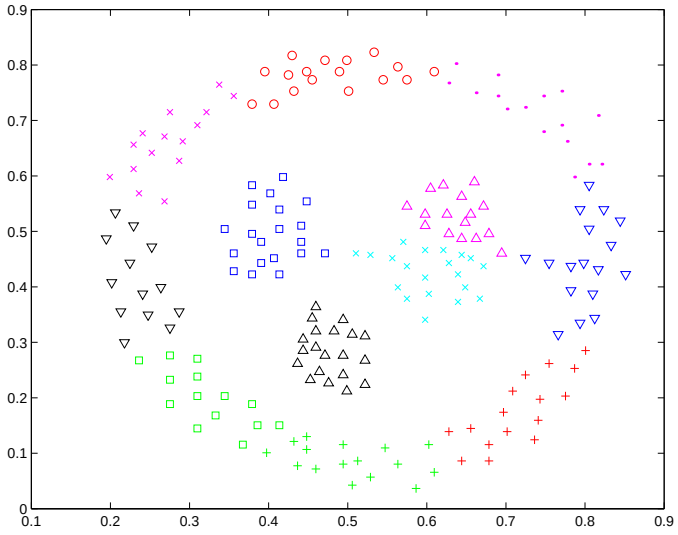
In the first step of the hybrid algorithm, I use a balanced version of the k-means algorithm presented in Chapter 5 to partition the data into fine granularity,

11 clusters for the *d4* dataset and 30 clusters for the *t4* dataset. When varying the number of clusters between 11 and 20 (for *d4*) and between 25 and 40 (for *t4*), it is observed that the final hybrid clustering results are stable. Fig. 6.6(a) & (b) show the balanced results. It can be seen that when clustering the data into fine enough granularity, one gets pure clusters (i.e. clusters that do not mix data from different natural clusters). The balance constraint helps restrict each cluster to be well defined (not empty or too large). The resulting stable, well-defined clusters form a good basis for the next step, hierarchical merging. Currently the number of clusters is user-selected; automatic methods for model selection will be investigated in the future.

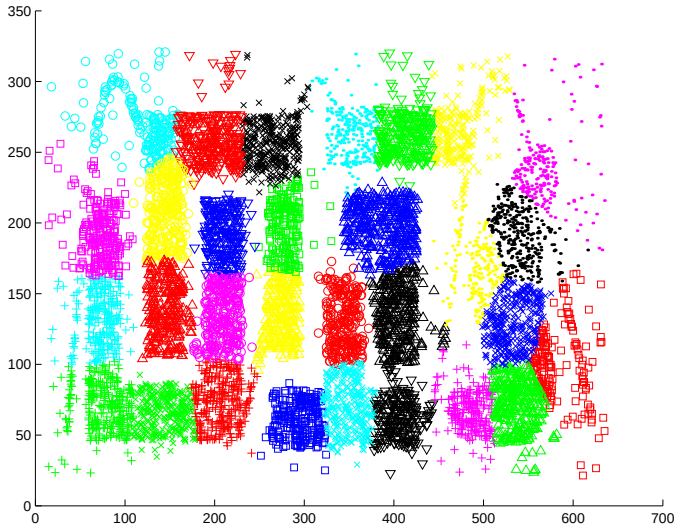
In the second step, I compute the cluster pairwise distances using the boundaryKL measure (3.16) with the parameter η (boundary area percentage) set to 0.1, and then apply single-link hierarchical meta-clustering to construct a meta-cluster hierarchy. I vary η between 0.05 and 0.2 and again observe that the final clustering results are stable. Fig. 6.7(a) & (b) show the meta-cluster hierarchies for the *d4* and *t4* datasets, respectively. From the hierarchies, it is evident that there are 4 clusters in *d4* and 6 in *t4*. When slicing the hierarchies at an appropriate granularity level, it can be seen from Fig. 6.8(a) & (b) that the hierarchical meta-clustering produces decent natural clusters. It is worth mentioning that the results are based on the boundaryKL distance which is more stable than the minKL measure. More experiments indicate that while the minKL measure can occasionally give reasonable results, the boundaryKL measure produces good results most of the time.

6.3.2 Results on Real Text Data

The *classic* and *k1b* datasets are used for empirical performance analysis on text data. Both datasets are contained in the datasets for the CLUTO software package

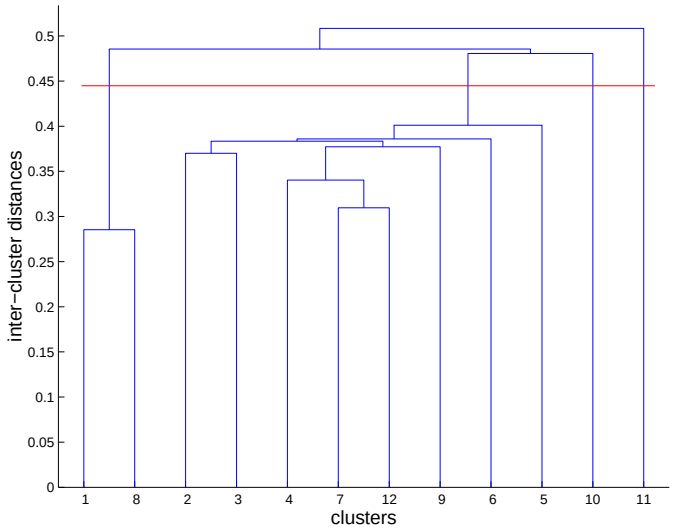


(a)

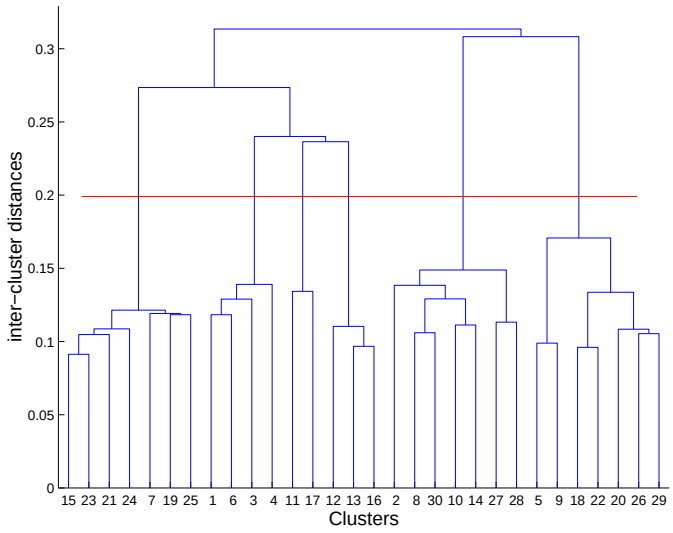


(b)

Figure 6.6: Results on the *d4* and *t4* datasets: balanced clustering of (a) *d4* dataset into 11 clusters and (b) *t4* dataset into 30 clusters.

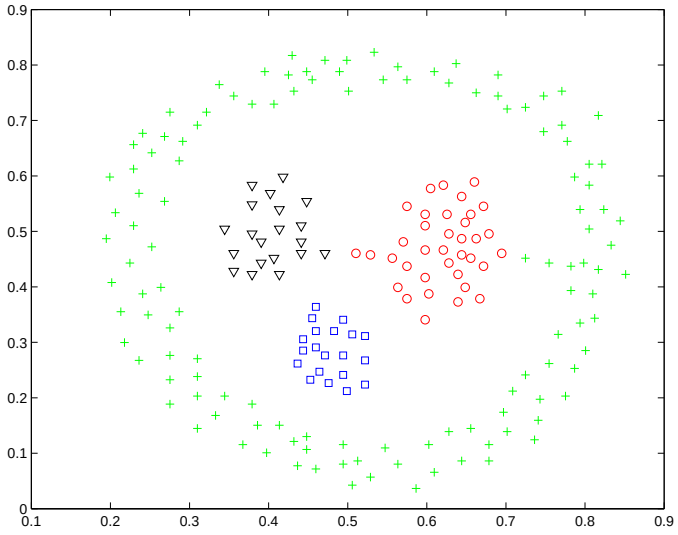


(a)

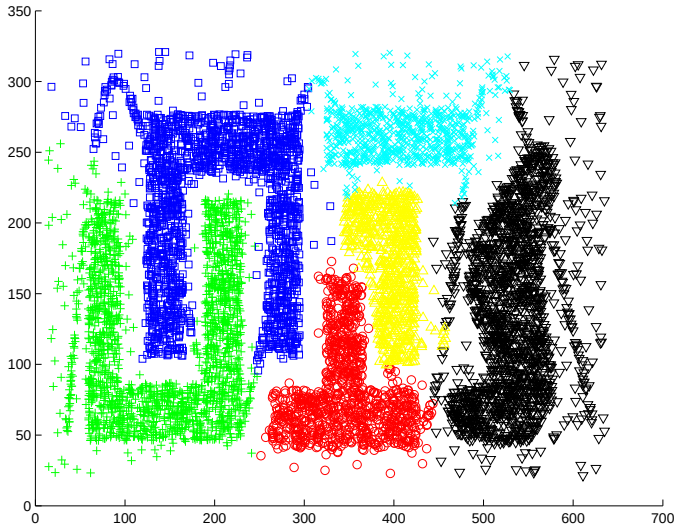


(b)

Figure 6.7: Results on the *d4* and *t4* datasets: meta-cluster hierarchy for (c) *d4* dataset and (d) *t4* dataset using boundaryKL distances.



(a)



(b)

Figure 6.8: Results on the *d4* and *t4* datasets: hybrid clustering results for (e) *d4* dataset in 4 clusters and (f) *t4* dataset in 6 clusters.

(Karypis, 2002) and have been described in section 4.3. The vMF models and log(IDF) weighted (and normalized) document vectors are used.

I compared the following algorithms for clustering the *classic* and *k1b* datasets:

- *k-vMFs:* model-based k-means with vMF models;
- *sk-vMFs:* stochastic mk-means with vMF models;
- *softvMFs:* soft clustering with mixture of vMF models[1];
- *b5-k-vMFs:* best (in terms of likelihood value) of every five runs of k-vMFs with each run starting from a (different) random initialization;
- *hier-k-vMFs:* hierarchical meta-clustering that uses k-vMFs in the flat clustering step and single-link hierarchical merging with the boundaryKL distance in the second step (In the result table, $6 \rightarrow 4$ means that we use 6 flat clusters in step 1 and merge them back to 4 in step 2.);
- *CLUTO:* graph partitioning algorithms built into the CLUTO toolkit (I used the *vcluster* algorithm in the toolkit with the default settings).

Table 6.1 show the NMI and average purity results, in the format of *average* ± 1 *standard deviation*, calculated over 10 runs. All partitional methods perform comparably, including CLUTO. For the *classic* dataset, the partitional methods are outperformed by the hybrid ones in terms of both NMI values and average purity results. For the *k1b* dataset, the partitional methods are outperformed by the hybrid ones in terms of average purity results, but not in terms of NMI values. This also shows that high average purity values do not necessarily translate into high NMI values. Different numbers of flat clusters are used and

[1] I used a constant κ for all clusters at each iteration (but κ gradually increases over 20 iterations from 20 up to 2000).

the results show that the number has a significant effect on the NMI results for hybrid vMF-based clustering. When an appropriate number is chosen, the hybrid clustering performs significantly better than the other methods. How to choose the number of flat clusters is a model selection problem, which is discussed further in the concluding chapter.

Table 6.1: Clustering results on *classic* and *k1b* datasets

classic			***k1b***		
	NMI	Average Purity		NMI	Average Purity
k-vMFs	$.54 \pm .03$	$.79 \pm .06$	k-vMFs	$.60 \pm .03$	$.78 \pm .04$
sk-vMFs	$.54 \pm .02$	$.78 \pm 0.03$	sk-vMFs	$.60 \pm .02$	$.79 \pm .03$
softvMFs	$.55 \pm .03$	$.79 \pm 0.04$	softvMFs	$.60 \pm .04$	$.79 \pm .03$
b5-k-vMFs	$.55 \pm .02$	$.82 \pm .02$	b5-k-vMFs	$.62 \pm .04$	$.80 \pm .04$
$(6 \rightarrow 4)$	$\mathbf{.63 \pm .03}$	$.84 \pm .03$	$(10 \rightarrow 6)$	$.59 \pm .06$	$\mathbf{.86 \pm .04}$
$(8 \rightarrow 4)$	$.63 \pm .04$	$\mathbf{.86 \pm .01}$	$(12 \rightarrow 6)$	$.59 \pm .07$	$.85 \pm .02$
$(10 \rightarrow 4)$	$.56 \pm .12$	$.86 \pm .02$	$(14 \rightarrow 6)$	$.56 \pm .03$	$.85 \pm .06$
CLUTO	$.54 \pm .02$	$0.73 \pm .03$	CLUTO	$\mathbf{.62 \pm .03}$	$.84 \pm .02$

6.3.3 Results on Synthetic and EEG Time-Series

Datasets

In this case study, I used three datasets—two synthetic (HMM-generated) datasets and a real EEG dataset.

The first synthetic dataset, *syn3*, contains three clusters and 60 sequences of length $T = 200$. The first 40 sequences are generated from two continuous HMM models (HMM1 and HMM2), 20 from each, same as in (Smyth, 1997). Both models have two hidden states and use the same priors and observation parameters. The priors are uniform and the observation distribution is univariate Gaussian with mean $\mu = 3$ and variance $\sigma^2 = 1$ for hidden state 1, and with mean $\mu = 0$ and variance $\sigma^2 = 1$ for hidden state 2. The state transition parameters

of HMM1 and HMM2 are $A_1 = \begin{bmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{bmatrix}$ and $A_2 = \begin{bmatrix} 0.4 & 0.6 \\ 0.6 & 0.4 \end{bmatrix}$, respectively. The remaining 20 sequences are composed of uniformly distributed random numbers, which can be seen as generated from a special HMM model that has only one state and uniform observation distribution. Fig. 6.9 shows one sample from each cluster. The second synthetic dataset, *syn3-50*, is simply a subset of the first one, containing only the first 50 time points of each sequence in *syn3*.

The EEG dataset, *EEG2*, is a subset of the *EEG12* dataset used in Section 5.3.4. For completeness, the data description is briefly repeated here. It is extracted from the UCI KDD Archive (Hettish and Bay, 1999) and contains measurements from an electrode (F4) on the scalp. There are 20 measurements from two subjects, a control subject and an alcoholic subject, 10 from each. Each measurement is sampled at 256Hz for 1 second, producing a sequence length of 256. Fig. 6.10 shows the 20 sequences. All values are scaled to be within $[-5, 5]$. The goal is to group the time series from the same subject together into one cluster. I model each cluster with a univariate HMM.

The number of hidden states in the HMMs is manually chosen. I used five hidden states for the synthetic datasets and eight for the EEG dataset. Also in the experiments I use classification accuracy as the evaluation criterion, assuming that the number of clusters is know a priori.

Clustering Results

I compare six HMM-based clustering algorithms on the three datasets described above. Four of them are partitional algorithms instantiated from the four generic model-based partitional clustering algorithms discussed in Section 3.2. By plugging in HMM models, I get HMM-based k-means (hk-means), soft hk-means, stochastic hk-means, and mixture-of-HMMs (moHMMs), respectively. The deriving details of the moHMMs approach using the EM algorithm can be found in Ap-

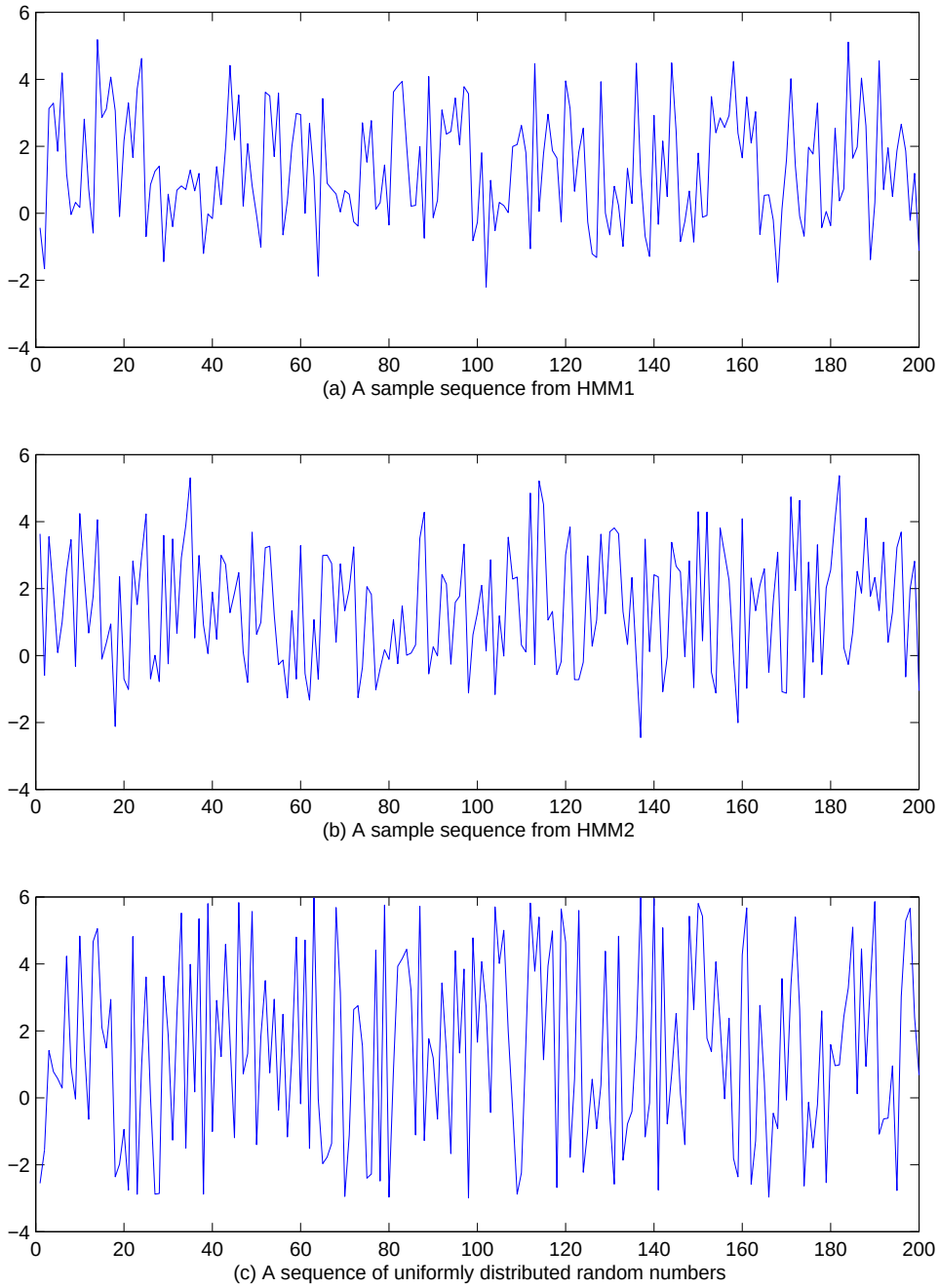


Figure 6.9: Synthetic time series from *HMM1*, *HMM2* and uniformly distributed numbers.

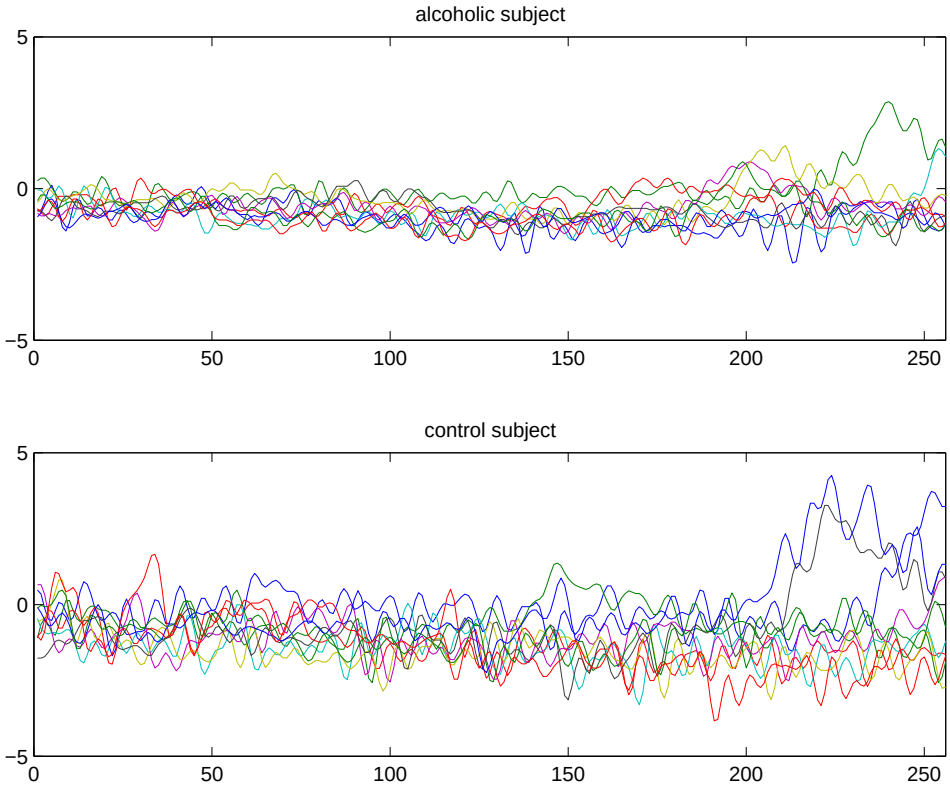


Figure 6.10: EEG time series for one alcoholic subject and one control subject.

pendix A. The complexity is $O(KNTN_s^2)$ for hk-means and stochastic hk-means and $O(K^2NTN_s^2)$ for soft hk-means and moHMMs, where T is the (maximum) length of sequences and N_s the number of hidden states in the HMM models. Two instantiated hybrid algorithms are

- *Hier-moHMMs:* the hybrid clustering algorithm (shown in Fig. 6.3) with the moHMMs algorithm used in the first step and the KL distance used for the second step;
- *Hier-hk-means:* the hybrid clustering algorithm (shown in Fig. 6.3) with the hk-means algorithm used in the flat clustering step and the KL distance used for the second step.

Clustering accuracy results are shown in Table 6.2. I run each algorithm 20 times (with different random initializations) and report the averages and standard deviations. Boldface font indicates the best performance. The results are presented in the form *average* ± 1 *standard deviation*. All four partitional methods perform comparably well and none is consistently better than the others across three datasets. Hybrid approaches clearly outperform partitional ones on all datasets. The hier-k-means is better than hier-moHMMs for short sequences (*syn3-50*).

Table 6.2: Clustering accuracy results on two synthetic datasets and one EEG dataset.

	syn3	*syn3-50*	*EEG2*
moHMMs	0.665 ± 0.127	0.71 ± 0.1	0.847 ± 0.122
hk-means	0.752 ± 0.176	0.76 ± 0.121	0.815 ± 0.142
soft hk-means	0.812 ± 0.158	0.798 ± 0.105	0.845 ± 0.12
stochastic hk-means	0.858 ± 0.195	0.742 ± 0.158	0.827 ± 0.14
hier-moHMMs	$\mathbf{0.905 \pm 0.154}$	0.832 ± 0.072	$\mathbf{0.885 \pm 0.083}$
hier-hk-means	0.86 ± 0.183	$\mathbf{0.882 \pm 0.031}$	0.852 ± 0.102

All methods have large standard deviations, indicating that the random initialization has a big effect on the clustering results, given the small data size. How to further reduce the initialization effect for HMM models remains an interesting task. Despite the high variances, I have run t-tests to measure the significance of the results, and observed that there are significant differences between the best hybrid result and the partitional ones.

6.3.4 Results on Gene Expression Time-Series Data

The advent of DNA microarray technology has provided an efficient way of measuring the expression levels of thousands of genes in a single experiment. As the number of microarray experiments keeps growing, one of the challenges is how to efficiently analyze the huge amount of expression data. For example, for the organism Saccharomyces cerevisiae (yeast), there are already over 700 experiments available on the internet for over 6000 genes. Furthermore, these experiments have been designed to uncover different aspects of gene behaviors and thus have different characteristics; some are independent gene mutation experiments and some are an ordered sequence of events, that is, a time series. Proper modeling of gene behaviors is essential to a meaningful analysis of the expression data.

Cluster analysis of gene expression data was introduced by Eisen et al. (1998) and has quickly attracted much research interest. The hierarchical approach used in (Eisen et al., 1998) is based on the Pearson correlation similarity measure between two genes, defined across all experiments. One version of this correlation similarity is exactly the cosine similarity between two normalized gene vectors. Model-based clustering has been used for gene expression analysis, but again is limited to Gaussian models (McLachlan et al., 2002; Yeung et al., 2001). In this section, I propose to use Markov chain models to capture the temporal properties of the expression time series.

Yeast Gene Expression Data

The cell cycle cdc-15 time series dataset downloaded from the Stanford genome database[2] contains 25 time points (measured every 20 minutes in a cell cycle experiment) for each gene. It is difficult to evaluate the clustering results of over 6000 genes given the large number of genes with unknown functions. A total of 222 genes were picked. All the picked genes have known function descriptions and participate in the pathways shown in Table 6.3. More information about the pathways and genes can be found on my web site.

Table 6.3: Pathways from which the 222 yeast genes are picked.

Regulatory Pathways	**Metabolic Pathways**
MAPK signaling pathway	Glycolysis/Gluconeogenesis
Cell cycle	Citrate cycle (TCA cycle)
Phosphatidylinositol signaling system	Pentose phosphate pathway
Second messenger signaling pathway	Galactose metabolism
	Oxidative phosphorylation
	Fatty acid biosynthesis (path 1)
	Fatty acid biosynthesis (path 2)
	Fatty acid metabolism
	Riboflavin metabolism
	Sterol biosynthesis
	Tryptophan metabolism
	Tetrachloroethene degradation

Markov Chain Models for Gene Expression

For gene expression data, one often cares more about whether a gene is over-expressed (i.e., up-regulated), under-expressed (i.e., down-regulated) or normally expressed, than how much it is up- or down-regulated. I thus discretize the expression data into five levels: 1–strongly down, 2–slightly down, 3–normal, 4–slightly

[2]http://genome-www.stanford.edu .

up, and 5–strongly up. I mean by "strongly" that the expressiveness of a gene is at least 2.5 times up or down from normal, and by "slightly" that the expressiveness is at least 1.2 times up or down from normal. After quantization, each gene vector becomes a discrete sequence with alphabet $\{1, 2, 3, 4, 5\}$. I used first-order Markov chains with five states.

Clustering Results

First, I use an MC-based k-means algorithm to get 20 flat clusters. The content of each cluster was examined by experts[3] from the Institute of Cell and Molecular Biology at The University of Texas at Austin.[4] As shown in Table 6.4, most of the clusters were well defined in terms of gene functionality. As an example, the contents of cluster 11 is shown in Table 6.5. It can be seen that most yeast genes in this cluster are related to mitosis. This indicates that the Markov chain model is an appropriate choice for this dataset. The contents of other clusters, as well as more results on the ~ 6000 yeast genes, can be found on my web site[5].

Table 6.4: Some coherent flat gene clusters

Cluster	Characteristics	Cluster	Characteristics
C1	DNA check point control	C8	Sporulation
C2	Cell cycle / mating / sporulation	C11	Mitosis
C3	Budding / sporulation	C12	Checkpoint
C4	Mating	C14	Heat production
C5	Cell devision	C18	Glucose metabolism
C6	Cell cycle arrest	C20	Mating
C7	Glucose metabolism		

In the hierarchical step, I calculate the mean vectors of all flat clusters, treat

[3]Data is not labeled, so one cannot use an extrinsic criterion for evaluation.

[4]I thank Xiaobin Wang, Zhenyu Zhang, and Yaning Wu for their help with the construction of the gene expression data and the analysis of the corresponding clustering results.

[5]http://www.ece.utexas.edu/∼szhong/gene.html.

Table 6.5: Contents of gene cluster 11

Name	GI#	Function Description
YER111C	6320957	"Involved in cell cycle dependent gene expression"
YCR084C	6319926	"General repressor of transcription (with Cyc8p); mediates glucose repression"
YFR028C	14318551	"Required for mitosis and sporulation"
YJL210W	6322250	"Required for peroxisome biogenesis"
YER125W	6320972	"Involved in ubiquitin-mediated protein degradation"
YMR043W	6323686	"Involved in cell-type-specific transcription and pheromone response"
YHR072W	6321863	"Carries out complex cyclization step of squalene to lanosterol in sterol biosynthesis pathway"
YER062C	6320905	"RHR2 (GPP1) encodes another DL-glycerol-3-phosphatase"
YAR071W	6319351	"Acid phosphatase, secreted"
YCR036W	10383797	"Ribokinase"
YIL106W	6681848	"Mps One Binder"
YLR006C	6323034	"Two-component signal transducer that with Sln1p regulates osmosensing MAP kinase cascade(suppressor of sensor kinase)"
YPL194W	6325062	"DNA damage checkpoint gene"
YBR136W	6319612	"Required for mitotic growth, DNA repair and mitotic recombination, regulates phosporylation of Rad53p, required for dmc1 arrest and meiotic recombination"
YLR377C	6323409	"Fructose-1,6-bisphosphatase"
YLR231C	6323261	"Biosynthesis of Nicotinic Acid"
YGR113W	14318436	"Duo1 And Mps1 interacting. Localized to intranuclear spindles and spindle pole bodies."
YOL001W	6324573	"Negative regulator of PHO81 and PHO5"
YDL003W	6320201	"Mitotic Chromosome Determinant; similar to S. pombe RAD21; may function in chromosome morphogenesis from S phase through mitosis"
YCL027W	10383763	"Cell-surface protein required for cell fusion"
YMR220W	6323876	"Involved in isoprene and ergosterol biosynthesis pathways"
YLR113W	6323142	"Osmoregulation"
YBR020W	6319494	"Haploid specific protein localized in the Golgi and plasma membrane"

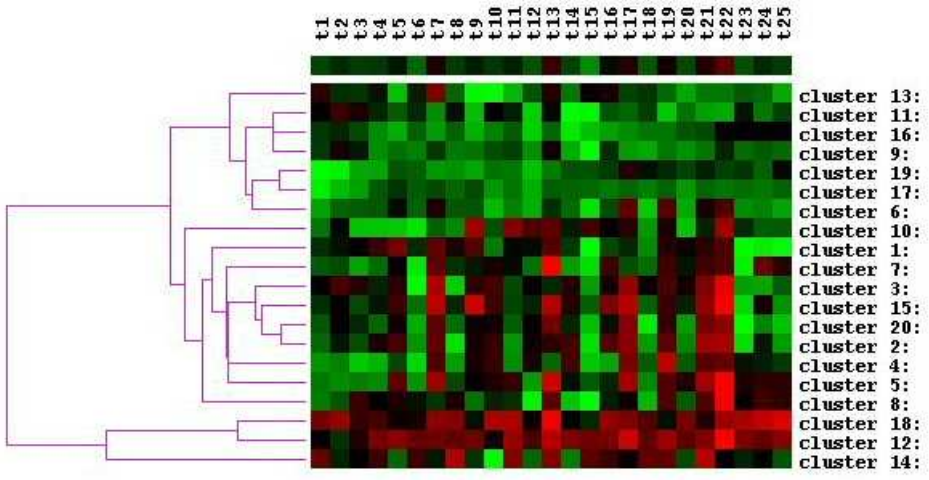


Figure 6.11: A meta-cluster hierarchy of 20 gene clusters.

them as 20 meta-objects, and use average-link hierarchical meta-clustering to form the meta-cluster hierarchy. Eisen's cluster analysis and visualization software[6] is used to produce the meta-cluster hierarchy (Fig. 6.11) which matches the flat cluster descriptions well. Furthermore, the results provide a useful visualization for understanding the relationship among the flat gene clusters.

6.4 Summary of Hybrid Clustering

I have proposed model-based hybrid algorithms that outperform model-based partitional clustering algorithms while possessing low computational complexities and providing useful hierarchical visualization and interactivity. The effectiveness of these model-based clustering algorithms has been highlighted by experimental comparisons in four different case studies. Hierarchical meta-clustering also points to some useful combinations of model-based partitional clustering with similarity-based hierarchical clustering.

[6]http://rana.lbl.gov/EisenSoftware.htm .

Chapter 7

Conclusion and Future Work

This dissertation approaches the problem of clustering complex data with model-based clustering, which the author argues is better suited to modern data mining applications. Compared to similarity-based clustering methods, model-based algorithms usually offer better interpretability, lower computational complexity, and easier online extensions. Many application-specific model-based clustering algorithms have been designed and successfully used but a unified treatment and analysis has not been developed before. This dissertation is motivated by the need to fill this void. It shows the benefits of having such a unified framework by both theoretical justifications and empirical comparative studies. This chapter first summarizes the main contributions of this dissertation in Section 7.1 and then outlines several potential future directions in Section 7.2.

7.1 Summary of Contributions

The main contributions of this dissertation include

1. *A unified framework for model-based clustering.* I have presented a unified framework for model-based clustering that provides a richer understanding of existing model-based clustering algorithms. The framework is applicable in any application domain for which good probabilistic models exist. Several

related model-based partitional clustering algorithms, including both soft and hard k-means type methods, have been analyzed in detail, and their relationships demonstrated from a deterministic annealing point of view. I have also designed a series of model-based hierarchical clustering algorithms, each of which is based on a different inter-cluster divergence measure. A clear distinction made between model-based and similarity-based hierarchical algorithms helps one gain a better understanding of existing hierarchical algorithms.

2. *An empirical case study on document clustering.* This case study not only investigates the suitability of different models to document clustering and compares the performance of different methods, but also represents a demonstration of the usefulness of the unified framework in Chapter 3. The construction of document clustering algorithms based on different models is easy and amounts to plugging in models into the generic algorithms discussed in Chapter 3. Also from the framework, it is clear that any model-based partitional clustering can be improved by imposing an annealing process on the data assignment (E-)step. This is confirmed by the experimental results shown in Section 4.5.

3. *Scalable, balanced model-based clustering.* Based on the general two-step (E-step and M-step) view of model-based partitional clustering in Section 3.2, I decompose the mk-means clustering problem into two subproblems. By applying a balance constraint in the data assignment step, a balanced data assignment subproblem is developed. The subproblem can be reduced to a linear programming problem that can be solved in $O(N^3)$ time, but a greedy heuristic is employed to approximately solve the subproblem in $O(N \log N)$ time. Experiments on a variety of datasets show that the balanced model-based clustering algorithm with maximum likelihood refinement can improve

clustering quality as well as the balance of clusters.

4. *Hybrid partitional-hierarchical clustering.* I have proposed hybrid model-based clustering algorithms that combine the advantages of model-based partitional algorithms and hierarchical algorithms. The hybrid algorithms can: (a) build a hierarchical view of data efficiently, based on iterative merging of fine-granularity clusters produced by model-based partitional methods; and (b) achieve high quality complex clusters by using suitable inter-cluster distances with the hierarchical meta-clustering. The effectiveness of the model-based hybrid clustering algorithms has been highlighted by experimental comparisons on various synthetic and real datasets.

7.2 Future Work

Future work can proceed in several directions.

Initialization strategies for model-based partitional clustering

All the model-based algorithms (except the deterministic annealing version) have a computational advantage over graph-partitioning based approaches but need better initialization strategies to generate consistent high-quality clustering results. Meila and Heckerman (2001) compared several initialization techniques and found none to be clearly better, so the quest for more effective techniques continues. Bradley and Fayyad (1998) employed sampling and meta-clustering (clustering of multiple solutions on sampled datasets) to refine initial cluster centroids. This technique deserves more investigation in the future.

The purpose of using a good initialization is to avoid some bad locally optimal solutions. The DA clustering can improve the model-based partitional

clustering algorithm by finding better local solutions. A second direction on improving the local solution of model-based algorithms is to tweak the clustering process. For example, local search has been employed by Dhillon et al. (2002) to improve the performance of the spherical k-means algorithm.

Online Extension of Model-based Clustering

Model-based partitional clustering algorithms can be made online, which is a desired feature in stream data mining applications. The competitive learning method, widely used in neural network literature, provides a way of constructing online k-means algorithms. It has been employed by Banerjee and Ghosh (2002a), Law and Kwok (2000), and Sinkkonen and Kaski (2001) for online model-based clustering of text documents, sequences, and gene expressions, respectively. Also online updates have been reported to work better than batch updates for both spherical k-means (Dhillon et al., 2001) and soft vMF-based clustering (Banerjee and Ghosh, 2002a).

More sophisticated online algorithms such as the self-organizing map (Kohonen, 1997) and the Neural-Gas algorithm (Martinetz et al., 1993) have a built-in annealing mechanism that can improve the chance of converging to better local solutions. As shown by the analysis in Section 3.2, these two algorithms can be easily extended to incorporate any probabilistic models for clustering complex data (e.g., sequences). This type of algorithms certainly deserve further investigation.

Model Selection: Choosing the Number of Clusters

A question I have not addressed is how to choose the final number of clusters, in either the partitional or the hierarchical/hybrid procedures. This is an old yet important problem for which a universally satisfactory answer has yet to be obtained.

Bayesian model selection techniques (Schwarz, 1978; Banfield and Raftery, 1993; Fraley and Raftery, 1998) have been investigated extensively. Most simple criteria such as BIC (Bayesian Information Criterion) or AIC (Akaike Information Criterion) either overestimate or underestimate the number of clusters, which severely limits their practical usability. Monte Carlo estimation of the posterior likelihood (Smyth, 1997) is more accurate but computationally expensive. Cross-validation methods can be effective when there is enough data; criteria such as the hold-out likelihood (Meila and Heckerman, 2001; Smyth, 1997) evaluated on a hold-out validation dataset often prove to be helpful.

It is often inappropriate to use model selection methods to find the number of clusters when models used are not a good description of the clusters. For example, the natural clusters in the synthetic 2-D datasets (Figs. 5.3 and 6.5) cannot be modeled by Gaussian distributions. Attempts to estimate the number of Gaussian distributions in a mixture of Gaussian distributions for clustering the data will lead to a very large number of clusters.

The use of hierarchical clustering alleviates the need to select the number of clusters because the user can interact with a hierarchical set of clusterings and choose the "best" one. Needless to say, in many clustering problems, only a human can give the best domain-specific judgment.

Deterministic annealing can be used to automatically determine the number of clusters for each temperature (Rose et al., 1993; Chakaravathy and Ghosh, 1996; Rose, 1998). Starting at a very high temperature (one cluster), one can form a top-down hierarchical structure of the data. Each split in the hierarchy represents a phase transition in the annealing process. This top-down hierarchical clustering using deterministic annealing may be extended to model-based clustering, thus avoiding the need of a model selection method.

Combining Model-based and Similarity-based Clustering

Recently, researchers have started to construct similarity measures from generative models. A model-based distance metric, called Riemannian distance, has been proposed by Tipping Tipping (1999) and Rattray Rattray (2000) for clustering. This metric, however, is computationally feasible only for Gaussian models. Another new model-based distance, pioneered by Amari Amari (1995) and Jaakkola and Haussler Jaakkola and Haussler (1999) and based on Fisher score (Amari, 1995), has been used in kernel classifier, e.g. support vector machines, to obtain better classification results (Jaakkola and Haussler, 1999; Jaakkola et al., 2000; Hofmann, 2000). But its effect on clustering has yet to be determined.

Another possible approach is to combine bipartite graph partitioning with model-based clustering based on Fig. 3.1. As mentioned in Section 3.1, the bipartite graph view provides a connection between model-based clustering and graph partitioning. One potential application is to use the bipartite graph partitioning techniques (Dhillon, 2001) in the second stage of the hybrid approach (Fig. 6.1), to directly obtain K clusters from an initial $K_0(> K)$ small clusters. Further research may reveal new, high quality clustering algorithms and insightful connections between generative and discriminative approaches.

Appendix A

Mixture of Hidden Markov Models

In this appendix I derive the detailed parameter re-estimation formula for the mixture-of-HMMs using the EM algorithm.

Let us start with one discrete observation sequence $X = \{x\}$ of length T. The mixture-of-HMMs model can be described as

$$P(X|\Lambda) = \sum_{k=1}^{K} \alpha_k p(x|\lambda_k), \text{ and } \sum_k \alpha_k = 1 \ ,$$

where K is the number of HMM components, $\lambda_k = \{\pi, A, B\}$ the k-th HMM component, α_k the mixture weight and $\Lambda = \{\alpha_k, \lambda_k\}_{k=1}^{K}$. To use the EM algorithm, let the missing data be (S, Y) and the complete data be (X, S, Y), where S is the set of hidden state sequences and $Y = \{1, ..., K\}$ the set of mixture component indices. Following the standard EM algorithm (Dempster et al., 1977; Blimes, 1998), I derive the EM update formulae for the mixture-of-HMMs as follows.

- *E-step:* The auxiliary function is

$$\begin{aligned} Q(\Lambda, \hat{\Lambda}) &= E[\log P(X, S, Y|\Lambda)|X, \hat{\Lambda}] \\ &= \sum_s \sum_y \log p(x, s, y|\Lambda) \cdot P(s, y|x, \hat{\Lambda}) \end{aligned}$$

$$
\begin{aligned}
&= \sum_s \sum_y \log\left(\alpha_y p(x, s|\lambda_y)\right) \cdot P(s, y|x, \hat{\Lambda}) \\
&= \sum_s \sum_y \log\left(\alpha_y \pi^{(y)}_{s_1} \prod_{t=1}^{T-1} a^{(y)}_{s_t s_{t+1}} \prod_{t=1}^{T} b^{(y)}_{s_t}(x(t))\right) \cdot P(s, y|x, \hat{\Lambda}) \\
&= \sum_s \sum_y \log \alpha_y \cdot P(s, y|x, \hat{\Lambda}) + \sum_s \sum_y \log \pi^{(y)}_{s1} \cdot P(s, y|x, \hat{\Lambda}) \\
&\quad + \sum_s \sum_y \sum_{t=1}^{T-1} \log\left(a^{(y)}_{s_t s_{t+1}}\right) \cdot P(s, y|x, \hat{\Lambda}) \\
&\quad + \sum_s \sum_y \sum_{t=1}^{T} \log\left(b^{(y)}_{s_t}(x(t))\right) \cdot P(s, y|x, \hat{\Lambda}) \ , \qquad \text{(A.1)}
\end{aligned}
$$

where s_t is the hidden state at time t, $\pi^{(y)}_{s1}$ the prior probability of hidden state s_1, $a^{(y)}_{s_t s_{t+1}}$ the transition probability of from state s_t to s_{t+1} and $b^{(y)}_{s_t}(x(t))$ the probability of observing $x(t)$ from state s_t. The superscript $^{(y)}$ means the y-th model. $\hat{\Lambda}$ is the current estimate of model parameters and Λ is the new estimate to be determined in the M-step.

- *M-step:* One can maximize $Q(\Lambda, \hat{\Lambda})$ with respect to α, π, a and b separately according to (A.1) in the E-step. First, the Lagrangian w.r.t. α is

$$
\mathcal{L}_\alpha = \sum_y \log \alpha_y \cdot P(y|x, \hat{\Lambda}) + \xi \left(\sum_y \alpha_y - 1\right) \ ,
$$

where ξ is the Lagrange constant. Let

$$
\partial \mathcal{L}_\alpha / \partial \alpha_y = 0 \ ,
$$

we get

$$
\frac{1}{\alpha_y} P(y|x, \hat{\Lambda}) + \xi = 0 \ .
$$

Multiply the equation by α_y and sum over y, we have

$$
\xi = -\sum_y P(y|x, \hat{\Lambda}) = -1
$$

and thus

$$
\alpha_y = -P(y|x, \hat{\Lambda})/\xi = P(y|x, \hat{\Lambda}) \ .
$$

By repeating the Lagrange optimization w.r.t. π, a and b, I obtain the following formulae:

$$\pi_j^{(y)} = P(s_1 = j|x, \hat{\lambda}_y) \ ,$$

$$a_{ij}^{(y)} = \frac{\sum_t P(s_t = i, s_{t+1} = j|x, \hat{\lambda}_y)}{\sum_t P(s_t = i|x, \hat{\lambda}_y)} \ ,$$

$$b_{jl}^{(y)} = \frac{\sum_t P(s_t = j, x(t) = l|x, \hat{\lambda}_y)}{\sum_t P(s_t = j|x, \hat{\lambda}_y)} \ .$$

For multiple observation sequences $X = \{x_1, x_2, ..., x_n\}$, I obtain the following EM parameter updates with a slightly more complex derivation:

$$\alpha_y = \frac{1}{N} \sum_{x \in X} P(y|x, \hat{\Lambda}) \ ,$$

$$\pi_j^{(y)} = \frac{\sum_{x \in X} P(y|x, \hat{\Lambda}) P(s_1 = j|x, \hat{\lambda}_y)}{\sum_{x \in X} P(y|x, \hat{\Lambda})} \ ,$$

$$a_{ij}^{(y)} = \frac{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t P(s_t = i, s_{t+1} = j|x, \hat{\lambda}_y)}{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t P(s_t = i|x, \hat{\lambda}_y)} \ ,$$

$$b_{jl}^{(y)} = \frac{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t P(s_t = j, x(t) = l|x, \hat{\lambda}_y)}{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t P(s_t = j|x, \hat{\lambda}_y)} \ .$$

The posterior component probability $P(y|x, \hat{\Lambda})$ used in above equations can be easily derived by Bayes rule as

$$P(y|x, \hat{\Lambda}) = \frac{\alpha_y p(x|\hat{\lambda}_y)}{\sum_{y'} \alpha_{y'} p(x|\hat{\lambda}_{y'})} \ .$$

Finally, to consider continuous observations, I use a mixture of Gaussians to model the observation distribution for each hidden state as described in Section 2.2. The corresponding EM updates for the parameters associated with continuous observation can be shown to be

$$c_{jl}^{(y)} = \frac{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t \gamma_{jl}^{(y)}(x, t)}{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t \gamma_j^{(y)}(x, t)} \ ,$$

$$\mu_{jl}^{(y)} = \frac{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t \left(\gamma_{jl}^{(y)}(x,t) \cdot x(t) \right)}{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t \gamma_j^{(y)}(x,t)} \ ,$$

$$\sigma_{jl}^{(y)} = \frac{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t \left(\gamma_{jl}^{(y)}(x,t) \cdot (x(t) - \mu_{jl}^{(y)})(x(t) - \mu_{jl}^{(y)})^T \right)}{\sum_{x \in X} P(y|x, \hat{\Lambda}) \sum_t \gamma_j^{(y)}(x,t)} \ ,$$

where $\gamma_{jl}^{(y)}(t)$ is the probability of the observation $x(t)$ being generated from Gaussian component l with the y-th model being at hidden state j and

$$\gamma_{jl}^{(y)}(x,t) = P(s_t = j|x, \hat{\lambda}_y) \cdot \frac{c_{jl}^{(y)} b_{jl}^{(y)}(x(t))}{b_j^{(y)}(x(t))} \ ,$$

$$\gamma_j^{(y)}(x,t) = \sum_l \gamma_{jl}^{(y)}(t) \ .$$

Bibliography

D. W. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton. Competitive learning algorithms for vector quantization. *Neural Networks*, 3(3):277–290, 1990.

E. L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*. Springer-Verlag, Berlin Heidelberg, 1990.

S. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1995.

K. M. Anstreicher. Linear programming in $o([n^3/\log n]l)$ operations. *SIAM Journal on Optimization*, 9(4):803–812, 1999.

A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on hyperspheres using expectation maximization. Technical Report TR-03-07, Department of Computer Sciences, The University of Texas at Austin, February 2003.

A. Banerjee and J. Ghosh. Frequency sensitive competitive learning for clustering on high-dimensional hyperspheres. In *Proc. IEEE Int. Joint Conf. Neural Networks*, pages 1590–1595, May 2002a.

A. Banerjee and J. Ghosh. On scaling up balanced clustering algorithms. In *Proc. 2nd SIAM Int. Conf. Data Mining*, pages 333–349, April 2002b.

J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49(3):803–821, September 1993.

Z. Bar-Joseph, G. Gerber, D. Gifford, and T. Jaakkola. A new approach to analyzing gene expression time series data. In *The 6th Annual Int. Conf. Research in Computational Molecular Biology*, pages 39–48, 2002.

L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1969.

Y. Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 2: 129–162, 1999.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

J. A. Blimes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical report, University of California at Berkeley, April 1998.

P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, Redmond, WA, 2000.

P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *Proc. 15th Int. Conf. Machine Learning*, pages 91–99, 1998.

M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 994–999, 1997.

I. V. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals and objects. In *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pages 140–149, 2000.

S. V. Chakaravathy and J. Ghosh. Scale based clustering using a radial basis function network. *IEEE Trans. Neural Networks*, 2(5):1250–61, Sept 1996.

D. Cutting, D. Karger, J. Pedersen, and J. W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proc. ACM SIGIR*, pages 318–329, 1992.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39 (1):1–38, 1977.

E. Dermatas and G. Kokkinakis. Algorithm for clustering continuous density HMM by recognition error. *IEEE Trans. Speech and Audio Processing*, 4(3): 231–234, May 1996.

I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pages 269–274, 2001.

I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*, pages 357–381. Kluwer Academic publishers, 2001.

I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proc. IEEE Int. Conf. Data Mining*, pages 131–138, Maebashi City, Japan, 2002.

I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14:755–763, 1998.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci.*, 95: 14863–14868, 1998.

S.-C. Fang and S. Puthenpura. *Linear Optimization and Extensions: Theory and Algorithms*. Prentice-Hall, 1993.

E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

C. Fraley. Algorithms for model-based Gaussian hierarchical clustering. *SIAM Journal on Scientific Computing*, 20(1):270–281, 1999.

C. Fraley and A. E. Raftery. How many clusters? Which clustering method? Answers via model-based analysis. *The Computer Journal*, 41(8):578–588, 1998.

A. S. Galanopoulos, R. L. Moses, and S. C. Ahalt. Diffusion approximation of frequency sensitive competitive learning. *IEEE Trans. Neural Networks*, 8(5): 1026–1030, September 1997.

J.-L. Gauvain and C.-H. Lee. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Trans. Speech and Audio Processing*, 2(2):291–298, April 1994.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell.*, 6(6):721–741, November 1984.

A. Gersho and R. M. Gray. *Vector Quantization and Signal Processing*. Kluwer Academic Publisher, Boston, MA, 1992.

A. B. Geva and D. H. Kerem. Brain state identification and forecasting of acute pathology using unsupervised fuzzy clustering of EEG temporal patterns. In H.-N. Teodorescu, A. Kandel, and L. C. Jain, editors, *Fuzzy and Neuro-Fuzzy Systems in Medicine*, chapter 3, pages 57–93. CRC Press, 1998.

J. Ghosh. Scalable clustering. In N. Ye, editor, *Handbook of Data Mining*, pages 341–364. Lawrence Erlbaum Assoc., 2003.

D. J. Hall and G. B. Ball. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12:153–155, 1967.

E. H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. WebACE: A web agent for document categorization and exploration. In *Proc. 2nd Int. Conf. Autonomous Agents*, pages 408–415, May 1998.

J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, 1975.

W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proc. ACM SIGIR*, pages 192–201, 1994.

S. Hettish and S. D. Bay. The UCI KDD archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science, 1999.

T. Hofmann. Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. In S. Solla, T. Leen, and K.-R. Muller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 914–920. MIT Press, 2000.

T. Hofmann and J. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(1):1–14, 1997.

T. Hofmann and J. Buhmann. Competitive learning algorithms for robust vector quantization. *IEEE Trans. Signal Processing*, 46(6):1665–1675, June 1998.

P. Indyk. A sublinear-time approximation scheme for clustering in metric spaces. In *Proc. 40th Symposium on Foundations of Computer Science*, pages 154–159, 1999.

T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1,2): 95–114, 2000.

T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 487–493. MIT Press, 1999.

A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

B.-H. Juang, S. E. Levinson, and M. M. Sondhi. Maximum likelihood estimation for multivariate mixture observations of Markov chains. *IEEE Trans. Information Theory*, 32(2):307–309, 1986.

B.-H. Juang and L. R. Rabiner. A probabilistic distance measure for hidden Markov models. *AT&T Technical Journal*, 64(2):391–408, 1985.

K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *Proc. IEEE Int. Conf. Data Mining*, pages 273–280, 2001.

A. Kalton, P. Langley, K. Wagstaff, and J. Yoo. Generalized clustering, supervised learning, and data assignment. In *Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pages 299–304, 2001.

S. Kamvar, D. Klein, and C. Manning. Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. In *Proc. 19th Int. Conf. Machine Learning*, pages 283–290, 2002.

R. Kannan, S. Vempala, and A. Vetta. On clusterings — good, bad and spectral. In *41st Annual IEEE Symp. Foundations of Computer Science*, pages 367–377, 2000.

G. Karypis. *CLUTO - A Clustering Toolkit*. Dept. of Computer Science, University of Minnesota, May 2002.

G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

M. Kearns, Y. Mansour, and A. Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proc. 13th Conf. Uncertainty in Artificial Intelligence*, pages 282–293, 1997.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

T. Kohonen. *Self-Organizing Map*. Springer-Verlag, New York, 1997.

T. Kohonen, S. Kaski, K. Lagus, J. Salojrvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Trans. Neural Networks*, 11(3):574–585, 2000.

A. Krogh. Hidden Markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

J. Kwon and K. Murphy. Modeling freeway traffic with coupled HMMs. Technical report, University of California at Berkeley, May 2000.

M. H. Law and J. T. Kwok. Rival penalized competitive learning for model-based sequence clustering. In *Proc. IEEE Int. Conf. Pattern Recognition*, pages 195–198, 2000.

C. Li and G. Biswas. Improving HMM clustering with Bayesian model selection. In *Proc. IEEE Int. Conf. Systems, Man and Cybernetics*, October 2000.

C. Li and G. Biswas. Applying the hidden Markov model methodology for unsupervised learning of temporal data. *International Journal of Knowledge-based Intelligent Engineering Systems*, 6(3):152–160, July 2002.

J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Information Theory*, 37:145–151, 1991.

S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Information Theory*, IT-28:129–137, March 1982.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Math. Statistics and Probability*, pages 281–297, 1967.

K. V. Mardia. Statistics of directional data. *J. Royal Statistical Society. Series B (Methodological)*, 37(3):349–393, 1975.

T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. "Neural-Gas" network for vector quantization and its application to time-series prediction. *IEEE Trans. Neural Networks*, 4(4):558–569, July 1993.

A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *AAAI Workshop on Learning for Text Categorization*, pages 41–48, 1998.

A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

G. McLachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

G. J. McLachlan, R. W. Bean, and D. Peel. A mixture model-based approach to the clustering of microarray expression data. *Bioinformatics*, 18:413–422, 2002.

M. Meila and D. Heckerman. An experimental comparison of model-based clustering methods. *Machine Learning*, 42:9–29, 2001.

T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 849–856. MIT Press, 2002.

K. Nigam. *Using Unlabeled Data to Improve Text Classification.* PhD thesis, School of Computer Science, Carnegie Mellon University, May 2001.

T. Oates, L. Firoiu, and P. R. Cohen. Clustering time series with hidden Markov models and dynamic time warping. In *IJCAI Workshop on Neural, Symbolic, and Reinforcement Methods for Sequence Learning*, Stockholm, Sweden, 1999.

J. Qian, M. Dolled-Filhart, J. Lin, H. Yu, and M. Gerstein. Beyond synexpression relationships: Local clustering of time-shifted and inverted gene expression profiles identifies new, biologically relevant interactions. *Journal of Molecular Biology*, 314:1053–1066, 2001.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.

M. Ramoni, P. Sebastiani, and P. Cohen. Bayesian clustering by dynamics. *Machine Learning*, 47:91–121, 2002.

W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(366):846–850, 1971.

E. Rasmussen. Clustering algorithms. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, 1992.

M. Rattray. A model-based distance for clustering. In *Proc. IEEE Int. Joint Conf. Neural Networks*, volume 4, pages 13–16, 2000.

I. Rezek and S. J. Roberts. Estimation of coupled hidden Markov models with application to biosignal interaction modeling. In *Proc. IEEE Int. Conf. Neural Network for Signal Processing*, volume 2, pages 804–813, 2000.

K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of IEEE*, 86(11):2210–2239, 1998.

K. Rose, E. Gurewitz, and G. C. Fox. Constrained clustering as an optimization method. *IEEE Trans. Pattern Anal. Machine Intell.*, 15:785–794, August 1993.

B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, 2001.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

J. Seo and B. Shneiderman. Interactively exploring hierarchical clustering results. *Computer*, 35(7):80–86, 2002.

J. Sinkkonen and S. Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14:217–239, 2001.

P. Smyth. Clustering sequences with hidden Markov models. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 648–654. MIT Press, 1997.

H. Stark and J. W. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, Boston, MA, August 2000.

A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

A. Strehl and J. Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing: Special Issue on Web Mining*, 15(2):208–230, 2003.

A. Strehl, J. Ghosh, and R. J. Mooney. Impact of similarity measures on web-page clustering. In *AAAI Workshop on AI for Web Search*, pages 58–64, July 2000.

M. Symons. Clustering criteria and multivariate normal mixtures. *Biometrics*, 37:35–43, 1981.

J. Tantrum, A. Murua, and W. Stuetzle. Hierarchical model-based clustering of large datasets through fractionation and refractionation. In *Proc. 8th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pages 239–246, 2002.

M. E. Tipping. Deriving cluster analytic distance functions from Gaussian mixture models. In *Proc. 9th IEEE Int. Conf. Artificial Neural Networks*, volume 2, pages 815–820, 1999.

A. K. H. Tung, R. T. Ng, L. V. D. Lakshmanan, and J. Han. Constraint-based clustering in large databases. In *Proc. 8th Int. Conf. Database Theory*, pages 405–419, 2001.

S. Vaithyanathan and B. Dom. Model-based hierarchical clustering. In *Proc. 16th Conf. Uncertainty in Artificial Intelligence*, pages 599–608, July 2000.

V. Vapnik. *Statistical Learning Theory.* John Wiley, New York, 1998.

J. H. Ward. Hierarchical groupings to optimize an objective function. *Journal of the American Statistical Association*, 58:234–244, 1963.

L. Xu, A. Krzyzak, and E. Oja. Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. *IEEE Trans. Neural Networks*, 4(4): 636–649, July 1993.

K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery, and W. L. Ruzzo. Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17(10):977–987, July 2001.

T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. ACM Int. Conf. Management of Data, Rec. 25*, volume 2, pages 103–114, 1996.

Y.-J. Zhang and Z.-Q. Liu. Self-splitting competitive learning: A new online clustering paradigm. *IEEE Trans. Neural Networks*, 13(2):369–380, March 2002.

Y. Zhao and G. Karypis. Criterion functions for document clustering: experiments and analysis. Technical Report #01-40, Department of Computer Science, University of Minnesota, November 2001.

S. Zhong and J. Ghosh. HMMs and coupled HMMs for multi-channel EEG classification. In *Proc. IEEE Int. Joint Conf. Neural Networks*, pages 1154–1159, May 2002a.

S. Zhong and J. Ghosh. A unified framework for model-based clustering and its applications to clustering time sequences. In *Intelligent Engineering Systems through Artificial Neural Networks (Proc. ANNIE)*. ASME Press, November 2002b.

S. Zhong and J. Ghosh. A comparative study of generative models for document clustering. In *SIAM Int. Conf. Data Mining Workshop on Clustering High Dimensional Data and Its Applications*, San Francisco, CA, May 2003a.

S. Zhong and J. Ghosh. Scalable, balanced model-based clustering. In *Proc. 3rd SIAM Int. Conf. Data Mining*, pages 71–82, San Francisco, CA, May 2003b.

S. Zhong and J. Ghosh. A unified framework for model-based clustering. Submitted to Journal of Machine Learning Research, July 2003c.

Vita

Shi Zhong was born on October 27, 1973 in Jingmen, a small city in Hubei Province, China. He received the B.S.E.E. from Huazhong University of Science and Technology (Wuhan, China) in 1994, and the M.S.E.E. from Tsinghua University (Beijing, China) in 1997. He spent two years in the Doctoral program in Electrical and Computer Engineering at the University of Minnesota, Twins Cities, before transferring to the University of Texas at Austin in 1999. His research interests include developing pattern recognition and data mining algorithms and applying them to solve various real world problems. He has published eight refereed technical papers in these areas. He is a student member of the IEEE and the IEEE computer society.

Permanent Address: 2201 Equestrian Trail, Austin, TX 78727, USA

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay and James A. Bednar.